КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизвації і інформаційних технологій
(факультет)

Кібербезпеки та комп'ютерної інженерії
(назва випускової кафедри)

КВАЛІФІКАЦІЙНА РОБОТА
ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему:

Розробка децентралізованої системи управління цифровою ідентичністю для IoT-пристроїв (DID for IoT)

Мочарська Валерія Тарасівна
(прізвище, ім'я та по батькові здобувача повністю)

Київ 2026 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизвації і інформаційних технологій
(факультет)

Кібербезпеки та комп'ютерної інженерії
(назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Делембовський М.М.
„___" _____20___року

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

Розробка децентралізованої системи управління цифровою ідентичністю для IoT-пристроїв (DID for IoT)
(назва)

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності.Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач Мочарська Валерія Тарасівна
(прізвище, ім'я та по батькові повністю)
125 Кібербезпека
(спеціальність)
Безпека інформаційних і комунікаційних систем
(освітня програма)
Група БІКС-22
Керівник Гуменний Д.О.
(прізвище та ініціали)

доцент кафедри кібербезпеки та комп'ютерної інженерії, кандидат технічних наук
(вчене звання, науковий ступінь)
Рецензент_____
(прізвище та ініціали)
Ідентичність підтверджую

Київ 2026

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій
Випускова кафедра: кібербезпеки та комп'ютерної інженерії
Ступінь вищої освіти: Бакалавр
Спеціальність: 125 Кібербезпека
Освітня програма: Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри
Делембовський М.М.
„___”_____20___року

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

Мочарська Валерія Тарасівна
(прізвище, ім'я та по батькові здобувача)

1. Тема роботи Розробка децентралізованої системи управління цифровою ідентичністю для IoT-пристроїв (DID for IoT)
затверджена наказом ректора КНУБА № 577/52-15/13.2/26 від «14» 05 2026року
2. Керівник роботи
Гуменний Д.О. доцент кафедри кібербезпеки та комп'ютерної інженерії, кандидат технічних наук, доцент
(прізвище, ім'я та по батькові, науковий ступінь, вчене звання)
3. Термін подання здобувачем роботи до захисту_____
4. Зміст пояснювальної записки за розділами:
Р. 1. ТЕОРЕТИЧНІ ОСНОВИ ДЕЦЕНТРАЛІЗОВАНИХ ІДЕНТИФІКАТОРІВ ТА ІОТ
Р. 2. ПРОЄКТУВАННЯ ДЕЦЕНТРАЛІЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ІДЕНТИЧНІСТЮ ДЛЯ ІОТ-ПРИСТРОЇВ
Р. 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ТЕСТУВАННЯ

7. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1 ТЕОРЕТИЧНІ ОСНОВИ ДЕЦЕНТРАЛІЗОВАНИХ ІДЕНТИФІКАТОРІВ ТА ІОТ	5.05.2026
Розділ 2 ПРОЄКТУВАННЯ ДЕЦЕНТРАЛІЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ІДЕНТИЧНІСТЮ ДЛЯ ІОТ-ПРИСТРОЇВ	20.05.2025
Розділ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ТЕСТУВАННЯ	29.05.2026
Остаточне оформлення роботи	9.06.2026
Направлення роботи для перевірки на плагіат	
Попередній захист роботи	12.06.2026
Направлення роботи на рецензування	

8. Дата видачі завдання 26.02.2026

Керівник Гуменний Д.О. _____

Здобувач Мочарська В.Т. _____

РЕЗЮМЕ (SUMMARY) *до кваліфікаційної випускової роботи здобувача*	ПІБ *здобувача українською та англійською мовами* *Мочарська Валерія Тарасівна* *Mocharska Valeriia*		
ЗВО	Київський національний університет будівництва і архітектури		
Тема *(українською та англійською)*	Розробка децентралізованої системи управління цифровою ідентичністю для IoT-пристроїв (DID for IoT). Development of a Decentralized Digital Identity Management System for IoT Devices (DID for IoT)		
Освітній ступінь	Бакалавр		
Факультет	Факультет автоматизвації і інформаційних технологій		
Випускова кафедра	Кафедра кібербезпеки та комп'ютерної інженерії		
Спеціальність	125 Кібербезпека		
Освітня програма	Безпека інформаційних і комунікаційних систем		
Керівник	Гуменний Д.О.		
Обсяг роботи:	*Поснювальна записка, стор.*	*Розділів*	*Презентація, кількість слайдів*
	93	3	10
Розділ 1	ТЕОРЕТИЧНІ ОСНОВИ ДЕЦЕНТРАЛІЗОВАНИХ ІДЕНТИФІКАТОРІВ ТА ІОТ		
Розділ 2	ПРОЄКТУВАННЯ ДЕЦЕНТРАЛІЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ІДЕНТИЧНІСТЮ ДЛЯ ІОТ-ПРИСТРОЇВ		
Розділ 3	ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ТЕСТУВАННЯ		
Висновки по роботі	Обґрунтовано вибір did:polygon та криптографічного стеку (Ed25519, ChaCha20, SHA-256). Розроблено архітектуру та програмний прототип, що забезпечує реєстрацію IoT-пристроїв, захист даних та публічну верифікацію. Запропонована система має переваги перед PKI: відсутність єдиної точки відмови, офлайн-верифікація, нульова вартість реєстрації, контроль власника		
Ключові слова: Keywords:	децентралізована ідентичність, DID, IoT, SSI, блокчейн, Polygon, IPFS, Ed25519, ChaCha20, кібербезпека, цифровий підпис, верифікація.		

	decentralized identity, DID, IoT, SSI, blockchain, Polygon, IPFS, Ed25519, ChaCha20, cybersecurity, digital signature, verification

Здобувач Мочарська В.Т. _____/ _____

Керівник Гуменний Д.О. _____/ _____

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці децентралізованої системи управління цифровою ідентичністю для IoT-пристроїв на основі технології децентралізованих ідентифікаторів (DID). У роботі проаналізовано недоліки існуючих централізованих систем ідентифікації (PKI, OAuth2, SAML), обґрунтовано вибір DID методу did:polygon та криптографічного стеку (Ed25519, ChaCha20-Poly1305, SHA-256). Розроблено архітектуру системи, яка складається з програмної симуляції IoT-пристрою на Node.js, блокчейну Polygon Amoy Testnet та децентралізованого сховища IPFS (Pinata). Реалізовано програмний прототип, що забезпечує реєстрацію пристрою, генерацію DID, криптографічний підпис та шифрування телеметричних даних, а також публічну верифікацію через блокчейн-експлорер. Проведено експериментальне тестування на чотирьох пристроях, підтверджено виконання вимог FR1-FR4. Виконано порівняльний аналіз з централізованою PKI, який підтвердив переваги запропонованого децентралізованого рішення.

Ключові слова: децентралізована ідентичність, DID, IoT, SSI, блокчейн, Polygon, IPFS, Ed25519, ChaCha20, кібербезпека, цифровий підпис, верифікація.

ABSTRACT

The qualification work is devoted to the development of a decentralized digital identity management system for IoT devices based on Decentralized Identifiers (DID) technology. The work analyzes the disadvantages of existing centralized identification systems (PKI, OAuth2, SAML), substantiates the choice of the did:polygon method and the cryptographic stack (Ed25519, ChaCha20-Poly1305, SHA-256). The system architecture is developed, consisting of a software simulation of an IoT device on Node.js, the Polygon Amoy Testnet blockchain, and a decentralized IPFS storage (Pinata). A software prototype is implemented that provides device registration, DID generation, cryptographic signing and encryption of telemetry data, as well as public verification through a blockchain explorer.

Experimental testing was conducted on four devices, confirming compliance with FR1-FR4 requirements. A comparative analysis with centralized PKI is performed, confirming the advantages of the proposed decentralized solution.

Keywords: decentralized identity, DID, IoT, SSI, blockchain, Polygon, IPFS, Ed25519, ChaCha20, cybersecurity, digital signature, verification.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

Скорочення	Розшифрування
API	Application Programming Interface (програмний інтерфейс застосування)
CA	Certificate Authority (центр сертифікації)
CID	Content Identifier (ідентифікатор вмісту в IPFS)
CRL	Certificate Revocation List (список відкликаних сертифікатів)
DID	Decentralized Identifier (децентралізований ідентифікатор)
DLT	Distributed Ledger Technology (технологія розподіленого реєстру)
EDR	Endpoint Detection and Response (виявлення та реагування на кінцевих точках)
EPC	Electronic Product Code (електронний код продукту)
Faucet	сервіс для отримання тестових токенів криптовалюти
HTTP	Hypertext Transfer Protocol (протокол передачі гіпертексту)
IdP	Identity Provider (провайдер ідентичності)
IMEI	International Mobile Equipment Identity (міжнародний ідентифікатор мобільного обладнання)
IoT	Internet of Things (Інтернет речей)
IPFS	InterPlanetary File System (міжпланетна файлова система)
ISO	International Organization for Standardization (Міжнародна організація зі стандартизації)
JSON	JavaScript Object Notation (формат обміну даними)
JSON-LD	JSON for Linking Data (JSON для зв'язування даних)
MAC	Media Access Control (керування доступом до середовища передачі)
NIST	National Institute of Standards and Technology (Національний інститут стандартів і технологій США)

OCSP	Online Certificate Status Protocol (протокол перевірки статусу сертифіката онлайн)
OIDC	OpenID Connect (протокол автентифікації)
ONS	Object Name Service (служба імен об'єктів)
PKI	Public Key Infrastructure (інфраструктура відкритих ключів)
POL	тестовий токен мережі Polygon
RAM	Random Access Memory (оперативна пам'ять з довільним доступом)
RFID	Radio Frequency Identification (радіочастотна ідентифікація)
ROM	Read Only Memory (постійний запам'ятовувальний пристрій)
RPC	Remote Procedure Call (виклик віддалених процедур)
SAML	Security Assertion Markup Language (мова розмітки тверджень безпеки)
SHA	Secure Hash Algorithm (алгоритм безпечного хешування)
SSI	Self-Sovereign Identity (самостійно керована ідентичність)
TLS	Transport Layer Security (протокол захищеного транспорту)
VC	Verifiable Credential (верифіковані облікові дані)
W3C	World Wide Web Consortium (Консорціум Всесвітньої павутини)
xIoT	eXtended Internet of Things (розширений Інтернет речей)

ВСТУП

Актуальність дослідження. Сучасні корпоративні та академічні інфраструктури вже тривалий час функціонують за межами класичного набору комп'ютерів та серверів. Сьогодні до корпоративних мереж підключаються десятки та сотні так званих "розширених" IoT-пристроїв (xIoT): розумні камери, конференц-системи, принтери, датчики клімату, системи контролю доступу та навіть побутова техніка. Проблема полягає в тому, що ці пристрої створюють велику проблемну зону з точки зору безпеки та ідентифікації. На відміну від традиційних IT-систем, більшість IoT-пристроїв не мають вбудованих механізмів для надійної, унікальної та довготривалої ідентифікації. Вони часто використовують заводські паролі, застарілі протоколи зв'язку, не підтримують стандартні засоби захисту (на кшталт антивірусів чи EDR-агентів) і не проходять регулярного оновлення прошивки. У результаті IT-відділи навіть не можуть точно сказати, скільки таких пристроїв підключено до їхньої мережі, не кажучи вже про те, щоб керувати їхніми життєвими циклами або перевіряти їхню справжність.

Саме ця відсутність надійної ідентифікації робить IoT-пристрої легкодоступною точкою входу для кібератак. Для досягнення своїх цілей порушникам не обов'язково долати надійно захищені серверні системи, достатньо знайти одну незахищену IP-камеру або розумний динамік зі стандартним паролем. Скомпрометувавши такий пристрій, атакуючий отримує повноцінну присутність у внутрішній мережі, звідки може проводити розвідку, викрадати облікові дані та пересуватися (lateral movement) до найцінніших IT-систем, тобто серверів баз даних, файлових сховищ, робочих станцій співробітників. Ключова особливість таких атак полягає в тому, що традиційні засоби безпеки "не бачать" компрометації IoT-пристрою. Оскільки реалізація атаки відбувається не шляхом запуску шкідливого файлу, а через використання легітимних функцій самого девайса або через його мережеву активність.

Традиційна інфраструктура відкритих ключів (PKI) виявляється непридатною для IoT через фундаментальні обмеження: технічну складність та проблеми масштабування, вразливість централізованих центрів сертифікації (CA), непридатність відкликання сертифікатів (CRL/OCSP) та високу вартість впровадження [15, с. 771-775]. Показовим є випадок компрометації CA DigiNotar у 2011 році, у результаті якого порушники згенерували більш ніж 500 фальшивих сертифікатів для доменів Google, Yahoo та інших сервісів, що змусило провідні браузери припинити довіру до цього CA, а сама компанія оголосила про банкрутство [10]. Детальний аналіз цих та інших недоліків наведено в розділі 1. Таким чином, центральна проблема сучасних IoT-мереж - це не просто відсутність оновлень або слабкі паролі, а принципова непридатність наявних централізованих підходів до управління ідентичностями (PKI, схеми логін/пароль) забезпечити масштабоване, децентралізоване та стійке до компрометації управління ідентичностями для мільярдів гетерогенних пристроїв. Кожен новий підключений датчик без власного децентралізованого ідентифікатора є потенційним каналом для компрометації мережі. Саме тому розробка децентралізованої системи управління цифровою ідентичністю на основі технології DID (DID for IoT) є не просто теоретичним завданням, а невідкладним практичним завданням для реалізації фундаментальних принципів гігієни кібербезпеки в сучасному цифровому середовищі.

Об'єкт дослідження - процеси автентифікації, авторизації та управління життєвим циклом ідентичностей у розподілених мережах Інтернету речей (IoT).

Предмет дослідження - методи, архітектурні рішення, криптографічні протоколи та програмні засоби для децентралізованого управління цифровою ідентичністю IoT-пристроїв на основі стандартів W3C Decentralized Identifiers (DID), адаптовані до обмежених обчислювальних ресурсів типових IoT-девайсів.

Мета дослідження - розробка прототипу децентралізованої системи управління цифровою ідентичністю для IoT-пристроїв на основі технології

децентралізованих ідентифікаторів (DID), яка забезпечує реєстрацію пристрою в блокчейні, криптографічний підпис телеметричних даних та публічну верифікацію через блокчейн- експлорер. яка симулює масштабовану, стійку до компрометації, конфіденційну та придатну для впровадження альтернативу традиційним централізованим рішенням (PKI), враховуючи обмеження апаратних платформ IoT.

Завдання дослідження. Завданням дослідження є:

1. Провести аналіз чинних систем ідентифікації, що застосовуються в IoT та виявити їхні недоліки.
2. Провести порівняльний аналіз DID методів та легковагових криптографічних алгоритмів, обґрунтувати вибір технологій для прототипу.
3. Сформулювати вимоги до системи.
4. Розробити архітектуру системи (компоненти, протоколи взаємодії, життєвий цикл DID, структури даних).
5. Реалізувати програмний прототип, що забезпечує створення DID, підпис та шифрування даних, реєстрацію в блокчейні та публічну верифікацію.
6. Провести тестування прототипу та виконати порівняльний аналіз з PKI.

Методологія дослідження. Методологія базується на поєднанні теоретичного аналізу, архітектурного проєктування та експериментальної верифікації.

У першому розділі застосовано метод систематичного огляду літератури для аналізу існуючих систем ідентифікації (PKI, OAuth2, SAML, Shibboleth), а також порівняльний підхід із метою ідентифікації їхніх недоліків (централізація, єдина точка відмови, проблеми масштабування).

У другому розділі застосовано метод порівняльного аналізу DID методів (did:key, did:ethr, did:indy, did:iota) для обґрунтування вибору розподіленого

реєстру. Метод криптографічного аналізу використано для порівняння легковагових алгоритмів (Ed25519, ChaCha20, XTEA, ASCON) за критеріями продуктивності та ресурсоспоживання. Архітектуру системи розроблено з використанням методів графічного моделювання (діаграма взаємодії та скінченний автомат, що описує життєвий цикл DID). Метод ранжування застосовано для вибору апаратної платформи (ESP32, Raspberry Pi, ARM Cortex-M) за критеріями вартості, енергоспоживання та наявності криптографічного прискорення.

У третьому розділі застосовано метод об'єктно-орієнтованого програмування для реалізації прототипу на Node.js. Метод натурного експерименту використано для перевірки працездатності системи: виконано реєстрацію тестових пристроїв у блокчейні Polygon Amoy Testnet, збережено отримані DID та хеші транзакцій. Метод порівняння застосовано для зіставлення розробленого рішення з централізованою PKI за критеріями централізації, єдиної точки відмови, вартості та часу анулювання сертифіката.

Структура роботи. Робота складається з трьох розділів.

У першому розділі розглянуто теоретичні основи децентралізованих ідентифікаторів (DID) та концепції самостійно керованої ідентичності (SSI). Виконано аналіз існуючих систем управління ідентичністю в IoT (PKI, OAuth2, SAML, Shibboleth) та виявлено їхні основні недоліки. Виконано порівняльне дослідження легковагових криптографічних алгоритмів, призначених для обмежених пристроїв.

У другому розділі сформульовано вимоги до децентралізованої системи управління ідентичністю. Виконано порівняльний аналіз DID методів (did:key, did:ethr, did:indy, did:iota) та обґрунтовано вибір розподіленого реєстру, криптографічного стеку та апаратної платформи. Розроблено архітектуру системи, діаграму послідовності реєстрації пристрою та скінченний автомат життєвого циклу DID.

У третьому розділі реалізовано програмний прототип на Node.js (програмна симуляція IoT-пристрою без використання фізичного обладнання), який забезпечує генерацію DID, криптографічний підпис телеметричних даних (статус безпеки: стан дверей, вікон, наявність руху), а також фіксацію пристрою в розподіленому реєстрі Polygon Amoy Testnet. Проведено експериментальне тестування: зареєстровано два тестових пристрої, отримано DID та хеші транзакцій, виконано публічну верифікацію через блокчейн-експлорер PolygonScan. Виконано порівняльний аналіз розробленого рішення з централізованою PKI.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ДЕЦЕНТРАЛІЗОВАНИХ ІДЕНТИФІКАТОРІВ ТА IOT

1.1 Проблеми ідентифікації в IoT: виклики та обмеження

Інтернет речей (IoT) є сучасною концепцією, що передбачає об'єднання об'єктів, або "речей", в єдину всесвітню мережу для взаємодії між собою та з людиною [3, с. 153]. Сьогодні кількість пристроїв, підключених до мережі, значно перевищує чисельність населення Землі й демонструє стійку тенденцію до збільшення, що робить критично важливим присвоєння кожному об'єкту унікальної адреси, забезпечення конфіденційності та безпеки при передачі даних [3, с. 153]. Незважаючи на це, досі не існує загальноприйнятого методу ідентифікації речей, який однаковою мірою відповідав би потребам як наявних, так і для нових пристроїв та додатків IoT. Ідентифікація є першою та ключовою вимогою в IoT, оскільки вона дозволяє розпізнавати мільярди гетерогенних об'єктів, забезпечувати дистанційне адміністрування пристроїв через глобальну мережу та встановлювати зв'язок об'єкт з інформацією, яку можна отримати з сервера [3, с. 155].

Аналіз дозволяє виділити наступні конкретні проблеми ідентифікації в IoT, які є фокусом даного дослідження.

Проблема 1. Відсутність вбудованих механізмів ідентифікації. Більшість IoT-пристроїв випускаються із заводськими налаштуваннями (стандартними паролями, відкритими портами, застарілими протоколами) і не мають апаратної або програмної підтримки для створення та управління власними криптографічними ідентифікаторами. Це унеможливлює їхню безпечну інтеграцію в корпоративну інфраструктуру без додаткових надбудов.

Проблема 2. Залежність від централізованих центрів сертифікації (CA). Традиційна інфраструктура відкритих ключів (PKI) на основі сертифікатів X.509 вимагає наявності довіреного центру сертифікації, який випускає та відкликає сертифікати. Проте в контексті IoT це породжує низку додаткових проблем: CA

стає єдиною точкою відмови (single point of failure), а компрометація CA (як у випадку DigiNotar у 2011 році [10]) унеможливлює довіру до всіх сертифікатів, випущених цим CA.

Проблема 3. Складність масштабування. Процес випуску, розповсюдження, оновлення та відкликання сертифікатів X.509 є громіздким і дорогим у масштабі мільярдів пристроїв. Крім того, механізми перевірки статусу сертифіката (CRL, OCSP) вимагають синхронного звернення до CA або репозиторію, що генерує надлишкове навантаження на мережу та призводить до зростання затримки [7; 22, с. 907-913].

Проблема 4. Неможливість офлайн-верифікації. У багатьох IoT-сценаріях пристрій може функціонувати в умовах обмеженої або періодичної мережевої з'ємності (наприклад, датчики у віддалених районах, на складах, у підвальних приміщеннях). Традиційні схеми, що вимагають синхронного звернення до CA для перевірки статусу сертифіката, у таких умовах непридатні.

Проблема 5. Відсутність контролю з боку власника. У централізованих моделях (наприклад, OAuth2, Google Home, Apple HomeKit) власник пристрою позбавлений безпосереднього впливу на ідентифікатор пристрою. Ідентифікатор створюється, зберігається та може бути анульований центральним провайдером без згоди власника. Це суперечить принципам цифрового суверенітету та ускладнює перенесення пристрою між різними платформами або організаціями.

Проблема 6. Ресурсні обмеження самих пристроїв. Типові IoT-пристрої (датчики, камери, контролери доступу) мають обмежену обчислювальну потужність, малий обсяг пам'яті (RAM/ROM) та працюють від батарей. Такі пристрої неспроможні реалізовувати ресурсомісткі криптографічні перетворення (наприклад, генерацію RSA-ключів або перевірку довгих ланцюжків сертифікатів) за прийнятний час та без значного енергоспоживання.

Для наочного уявлення перелічені проблеми зведено в таблицю 1.1.

Таблиця 1.1 - Основні проблеми ідентифікації в IoT

№	Проблема	Короткий опис
1	Відсутність вбудованих механізмів	Пристрої не мають апаратної або програмної підтримки для створення криптографічних ідентифікаторів
2	Залежність від централізованих CA	Єдина точка відмови, ризик компрометації CA
3	Складність масштабування	Висока вартість та громіздкість процесів випуску, оновлення, відкликання сертифікатів
4	Неможливість офлайн-верифікації	Вимагається синхронне звернення до CA або OCSP-респондера
5	Відсутність контролю власника	Ідентифікатор належить провайдеру, а не власнику пристрою
6	Ресурсні обмеження пристроїв	Обмежена пам'ять, обчислювальна потужність, енергоспоживання

Перспектива вирішення. Подолання описаних недоліків є можливим через перехід від централізованих моделей ідентифікації до децентралізованих, де:

- ідентифікатор (DID) створюється самим пристроєм без участі центрального провайдера,
- публічний ключ публікується в розподіленому реєстрі (блокчейні), що унеможливлює його підробку або блокування,
- перевірка статусу ідентифікатора виконується без синхронного звернення до CA,
- власник пристрою (його гаманець) отримує виключний контроль над DID.

1.2 Аналіз централізованих та федеративних моделей ідентифікації

Для розуміння переваг децентралізованої ідентифікації необхідно

проаналізувати існуючі централізовані та федеративні моделі, їхні сильні сторони та, головне, недоліки в контексті IoT-середовищ.

У централізованих моделях існує єдиний довірений посередник - провайдер ідентичності (IdP) або центр сертифікації (CA), який створює, зберігає, підтверджує та відкликає ідентифікатори. Клієнт (IoT-пристрій) надсилає запит до цього проміжного елемента з метою підтвердження справжності або отримання сертифіката.

OAuth2 є протоколом авторизації, який дозволяє додаткам отримувати обмежений доступ до ресурсів без розкриття облікових даних. Для IoT це створює проблеми: токени доступу потребують постійної мережевої верифікації. Дослідження IIETA [4] показує, що система на основі OAuth2 для 1000 одночасних IoT-пристроїв демонструє час токена до 500 мс, що є прийнятним для помірних навантажень, однак при збільшенні кількості пристроїв затримки зростають через необхідність кожного разу звертатися до центрального сервера. Крім того, традиційні схеми OAuth2 не передбачають вбудованого механізму для підпису даних самим пристроєм, що вимагає довіри до центрального посередника для виконання підтвердження автентичності повідомлень.

SAML (Security Assertion Markup Language) є стандартом обміну даними про автентифікацію між доменами, заснованим на XML. Його ключовий недолік для IoT - значний розмір повідомлень. Згідно з аналітичними даними 2026 року, типовий SAML-асершн займає від 2 до 8 кілобайт, а в реальних умовах (з додатковими сертифікатами та атрибутами) сягає 11 кілобайт після base64-кодування [9]. Для порівняння, сучасний аналог OIDC (OpenID Connect) формує токен розміром від 0,8 до 1,5 кілобайт - у 4-5 разів менше [9]. Додатково, SAML потребує обробки XML-документів з просторами імен (namespaces), канонізації та перевірки XML-підписів, що створює значне обчислювальне навантаження. Як зазначено в технічній дискусії Shibboleth, проблеми з парсингом SAML-асершнів можуть виникати навіть у потужних серверних середовищах при

високих навантаженнях [3; 5], не кажучи вже про обмежені IoT-пристрої. Експериментальні дані показують, що верифікація SAML-асершна на сучасному сервері займає 18-25 мілісекунд, тоді як перевірка JWT-токена OIDC - лише 1-2 мілісекунди, тобто у 10-20 разів швидше [9].

PKI (Public Key Infrastructure) на основі сертифікатів X.509 є фундаментом безпеки для багатьох протоколів (TLS, HTTPS, MQTT). Однак стандартний X.509-сертифікат займає близько 200-220 байт лише для ключової інформації, а з усіма полями - значно більше [6]. Зберігання навіть одного такого сертифіката в захищеній пам'яті обмеженого пристрою може потребувати кількох EEPROM-слотів [6]. Механізми перевірки статусу сертифікатів також мають кількісні обмеження: час відповіді OCSP-сервера згідно з документацією Huawei IoT становить до 5 секунд при таймауті, а кеш OCSP-відповіді оновлюється лише кожні 24 години [2]. CRL-файли (списки відкликаних сертифікатів) популярних CA мають середній розмір близько 2 мегабайт, тоді як усереднена тривалість відповіді OCSP дорівнює приблизно 430 мілісекунд [8]. Для пристрою, який надсилає дані щохвилини, такі затримки та обсяги трафіку можуть бути критичними.

Федеративні моделі дозволяють різним організаціям обмінюватися інформацією про ідентичність через попередньо узгоджені довірчі відносини. Однак вони успадковують проблеми SAML (великий розмір асершнів, XML-обробка) і додають власні. Експериментальні дані з навантажувального тестування Shibboleth IdP показують, що при високих навантаженнях процес shibd досягає 99% використання CPU, а час відповіді зростає з 50 мілісекунд до понад однієї хвилини за дуже короткий проміжок часу [5]. Такі обставини унеможливлюють застосування федеративних архітектур у масштабних підключеннях IoT-пристроїв, особливо в сценаріях, де потрібне миттєве підтвердження ідентичності.

Для наочного кількісного порівняння розглянутих моделей нижче наведено

зведену таблицю 1.2.

Таблиця 1.2 - Кількісне порівняння централізованих та федеративних моделей ідентифікації для IoT

Критерій	Централізовані	Федеративні
Розмір токена/асершна	SAML: 2-8 КБ (до 11 КБ з кодуванням) X.509: ~200-220 Б	SAML: ті ж 2-8 КБ (успадковано)
Час верифікації	OIDC: 1-2 мс SAML: 18-25 мс	XML-верифікація: до 20+ мс
Час перевірки статусу	OCSP: до 5 с таймаут, кеш 24 год CRL: ~2 МБ, OCSP: ~430 мс	Синхронне звернення до IdP
Єдина точка відмови	Так (IdP або CA)	Так (IdP домашньої організації)
CPU навантаження при пікових навантаженнях	OAuth2 для 1000 пристроїв: до 500 мс затримки	shibd: 99% CPU, час відповіді з 50 мс до >1 хв

Проведений кількісний аналіз показує, що централізованим та федеративним підходам до ідентифікації притаманні фундаментальні недоліки, які роблять їх непридатними для масового впровадження в IoT-середовищах. Особливо критичними є:

- великий розмір даних (SAML до 11 КБ проти 1 КБ OIDC [9]),
- значні затримки (верифікація SAML 18-25 мс проти 1-2 мс OIDC [9]),
- високе CPU навантаження (до 99% для Shibboleth [5]),
- наявність єдиної точки відмови (CA або IdP),
- необхідність синхронної перевірки (OCSP до 5 с [2]).

Саме з цієї причини перспективним вектором розвитку є перехід до

децентралізованих моделей на основі технології Self-Sovereign Identity (SSI) та децентралізованих ідентифікаторів (DID), які усувають зазначені недоліки.

1.3 Немережеві, апаратні та локальні методи ідентифікації та їхні обмеження

Поряд із програмними системами ідентифікації (OAuth2, SAML, PKI), які було розглянуто в підрозділі 1.2, існує велика група традиційних методів ідентифікації об'єктів. Переважна кількість цих методів виникла на початкових стадіях розвитку IoT і продовжують використовуватись у специфічних сценаріях. Однак жоден з них не забезпечує криптографічної верифікації в поєднанні з децентралізацією, що необхідно для сучасних вимог безпеки.

Немережеві методи (RFID, штрих-коди, EPC).

Радіочастотна ідентифікація (RFID) використовує радіохвилі для автоматичної ідентифікації об'єктів, до яких прикріплені мітки. Вважається однією з ключових технологій в IoT, особливо в логістиці та складському обліку. Штрих-коди є оптичним представленням машинозчитувальних міток, які записують дані про продукт [3, с. 158]. EPC (Electronic Product Code) коди забезпечують унікальну ідентифікацію об'єктів.

Ключові обмеження, що перешкоджають застосуванню в системах адміністрування ідентичністю IoT:

- Відсутність криптографічного захисту - RFID-мітки не мають вбудованих механізмів для цифрового підпису або шифрування даних, що робить їх вразливими до клонування та перехоплення.
- Обмежений радіус дії - пасивні RFID-мітки працюють на відстані до кількох метрів, активні - до кількох десятків метрів, що унеможливлює глобальну ідентифікацію через Інтернет.
- Статичність - EPC-код, як правило, незмінний і не підтримує оновлення або ротацію ключів.
- Централізоване зберігання - дані про мітки зберігаються в центральних базах

даних (наприклад, ONS - Object Name Service), що створює єдину точку відмови.

Апаратні ідентифікатори (MAC-адреса, IMEI, серійний номер).

Значна частина мережевих пристроїв оснащена апаратно фіксованими неповторними ідентифікаторами: MAC-адресу (для мережевих інтерфейсів), IMEI (для мобільних пристроїв), серійний номер (виробника). Ці ідентифікатори часто використовуються для автентифікації в корпоративних мережах (наприклад, через білі списки MAC-адрес).

Обмеження:

- Легкість підробки - MAC-адресу можна змінити програмно (MAC spoofing). Згідно з дослідженнями, більшість операційних систем дозволяють змінювати MAC-адресу без спеціальних прав [5].
- Статичність - IMEI та серійний номер неможливо змінити, що унеможливлює ротацію ключів при компрометації.
- Відсутність криптографічної верифікації - сам по собі апаратний ідентифікатор не доводить, що повідомлення надіслане саме тим пристроєм (необхідний цифровий
- підпис).
- Проблеми конфіденційності - MAC-адреси та IMEI можуть бути використані для відстеження переміщень пристрою без згоди власника [5].
- Мережеві адреси (IPv6, 6LoWPAN).IP-адреси, особливо у версії IPv6, використовуються для логічної ідентифікації кінцевого обладнання на мережевому рівні. Протокол IEEE 802.15.4 та технологія 6LoWPAN застосовуються в IoT через низьке енергоспоживання та здатність працювати з великою кількістю сенсорних пристроїв [3, с. 157]. Специфікація IPv6 визначає довжину адреси як 128 біт, що дозволяє адресувати до 2^128 пристроїв.

Однак IP-адреса не є ідентифікатором пристрою в криптографічному сенсі:

- Динамічність - адреса може змінюватися при переміщенні пристрою між мережами (DHCP, SLAAC).
- Відсутність криптографічного зв'язку - IP-адреса не пов'язана з публічним ключем пристрою, тому її неможливо використати для перевірки цифрового підпису.
- Перевантаження ролей - в IPv6 часто намагаються застосовувати одну й ту саму адресу одночасно для розпізнавання пристрою та маршрутизації трафіку, що створює конфлікти безпеки та конфіденційності (наприклад, використання постійної частини адреси на основі MAC, що дозволяє відстежувати пристрій).

Для наочного порівняння немережевих, апаратних та локальних методів нижче подано узагальнену таблицю 1.3.

Таблиця 1.3 - Порівняння немережевих, апаратних та локальних методів ідентифікації для IoT

Метод	Крипто-верифікація	Децентралізація	Оновлення/ ротація	Глобальна адресація	Контроль власника
RFID мітки	Ні	Централізована база	Статичний код	Лише локально	Ні
Штрих-коди	Ні	Централізована база	Статичний	Лише локально	Ні
EPC	Ні	ONS (центральний)	Статичний	Через ONS	Ні
MAC-адреса	Ні	Локальна таблиця	Ні	Лише в межах LAN	Ні

IMEI	Ні	Бази виробників	Ні	Глобально	Ні
IPv6-адреса	Ні	Так (глобальна)	Так (DHCP/SLAAC)	Глобально	Частково

Продовження табл. 1.3

Проведений аналіз показує, що традиційні методи ідентифікації (RFID, штрих-коди, EPC, MAC-адреси, IMEI, IPv6) мають фундаментальні обмеження, які роблять їх непридатними для побудови децентралізованої системи управління ідентичністю IoT-пристроїв:

- Відсутність криптографічної верифікації - жоден із проаналізованих підходів не гарантує можливість підтвердження справжності повідомлення через цифровий підпис.
- Централізація - більшість методів покладаються на центральні бази даних (ONS, реєстри виробників, ARP-таблиці), що створює єдину точку відмови та ризик цензури.
- Статичність - апаратні ідентифікатори неможливо оновити при компрометації (на відміну від криптографічних ключів, які можна ротувати).
- Відсутність контролю власника - ідентифікатор належить виробнику (IMEI) або мережевому адміністратору (MAC, IP), а не власнику пристрою.

Отже, для створення актуальної системи адміністрування ідентичностями слід використовувати криптографічні ідентифікатори (публічні ключі) в поєднанні з децентралізованим реєстром (блокчейном) та механізмом цифрового підпису. Такий підхід реалізується в технології децентралізованих ідентифікаторів (DID).

1.4 Концепція самостійно керованої ідентичності (SSI)

У відповідь на виявлені недоліки централізованих (п. 1.2) та традиційних (п. 1.3) методів ідентифікації виникла нова парадигма - самостійно керована

ідентичність (Self-Sovereign Identity, SSI). Як зазначає Крістофер Аллен у своїй основоположній роботі 2016 року, SSI базується на десяти ключових принципах [3]:

1. Існування (Existence) - суб'єкт ідентичності повинен мати незалежне існування, прив'язане до жодної зовнішньої системи.
2. Контроль (Control) - власник володіє абсолютним правом керування власним цифровим профілем.
3. Доступ (Access) - суб'єкт повинен мати доступ до власних даних та мати можливість їх переглядати.
4. Прозорість (Transparency) - системи управління ідентичністю мають бути відкритими та зрозумілими.
5. Персистентність (Persistence) - ідентичність повинна бути довготривалою, а не тимчасовою.
6. Портативність (Portability) - ідентичність повинна бути доступною та використовуватися в різних системах і сервісах.
7. Сумісність (Interoperability) - ідентичність повинна працювати з різними технологіями та платформами.
8. Згода (Consent) - використання даних можливе лише за явною згодою власника.
9. Мінімізація (Minimization) - розкриття лише тих даних, які необхідні для конкретної взаємодії.
10. Захист (Protection) - система повинна гарантувати права суб'єкта та захист від несанкціонованого доступу.

Серед перелічених принципів для управління ідентичністю IoT-пристроїв найважливішими є: контроль (власник пристрою керує його ідентифікатором), портативність (ідентифікатор не прив'язаний до виробника або платформи) та мінімізація (можливість підтвердити стан пристрою без розголошення надлишкової інформації).

SSI є децентралізованою моделлю цифрової ідентичності, в якій суб'єкт (особа,

організація або пристрій) має виключне право власності на свою ідентичність без необхідності покладатися на централізовану третю сторону [6, с. 111].

Архітектура SSI базується на трьох основних компонентах [6, с. 112]:

1. Децентралізований ідентифікатор (DID) - унікальний рядок, який слугує постійним ідентифікатором суб'єкта. DID створюється самим суб'єктом без участі центрального органу.
2. Верифіковані облікові дані (Verifiable Credentials, VC) - криптографічно підписані твердження емітента про суб'єкта (наприклад, «пристрій пройшов сертифікацію»).
3. Цифровий гаманець - програмний або апаратний компонент, який зберігає DID, VC та криптографічні ключі.

Взаємодія між цими компонентами дозволяє суб'єкту створювати DID та пару ключів, публікувати публічний ключ у розподіленому реєстрі, отримувати від емітента підписані VC, а потім надавати верифікатору криптографічний доказ без розкриття зайвої інформації та без синхронного звернення до емітента.

Концепція SSI впроваджує якісно іншу парадигму регулювання цифровою ідентичністю, який усуває фундаментальні недоліки централізованих моделей: відсутність єдиної точки відмови, можливість верифікації без синхронного звернення до емітента, повний контроль з боку власника. Ключовим елементом SSI, який безпосередньо реалізує ці властивості, є децентралізований ідентифікатор (DID).

1.5 Децентралізовані ідентифікатори (DID) та DID-документ

Децентралізований ідентифікатор (DID) є ключовим компонентом архітектури SSI та стандартом W3C [20]. DID - це простий текстовий рядок, який слугує постійним, унікальним та верифікованим ідентифікатором для будь-якого суб'єкта (особи, організації, пристрою або цифрового об'єкта). На відміну від традиційних ідентифікаторів (наприклад, email-адреси або логіна), DID

створюється самим суб'єктом без участі центрального органу (провайдера, реєстратора, центру сертифікації).

Структура DID. DID має уніфікований формат, який складається з трьох обов'язкових частин [20], що зображені на Рисунку 1.1

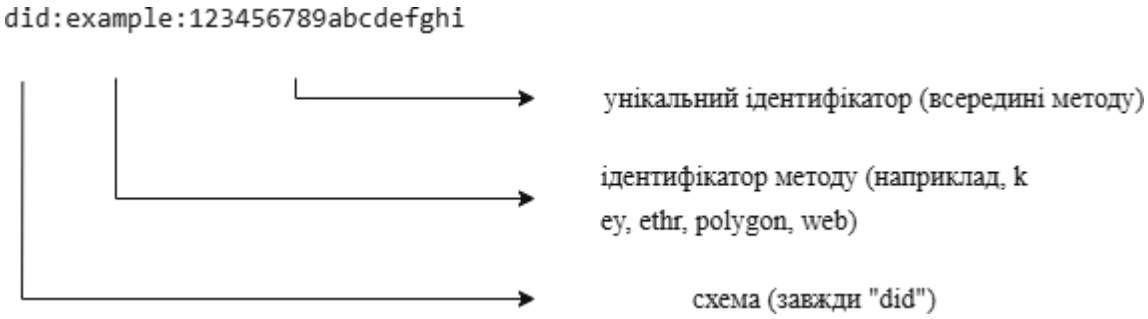


Рисунок 1.1 - Структура децентралізованого ідентифікатора (DID) відповідно до стандарту W3C

Схема «did» виступає сталим початковим фрагментом для будь-яких децентралізованих ідентифікаторів. Ідентифікатор методу визначає конкретну технологію розподіленого реєстру або механізм резолвінгу (наприклад, key, ethr, polygon, iota). Унікальний ідентифікатор є метод-специфічним рядком, що забезпечує беззаперечне розпізнавання суб'єкта в рамках визначеного методу.

З кожним DID асоціюється DID-документ - структурований набір даних (у форматі JSON або JSON-LD), який публікується в розподіленому реєстрі (блокчейні) та містить криптографічну інформацію, необхідну для верифікації суб'єкта [20].

Основні поля DID-документа:

- id - власне DID, якому відповідає цей документ.
- controller (необов'язкове) - DID суб'єкта, який має право вносити зміни до документа (дозволяє делегувати управління).
- verificationMethod - набір криптографічних матеріалів (публічних ключів) разом із метаданими, які визначають, як ці ключі можуть бути використані для верифікації.
- authentication - визначає, які саме процедури верифікації застосовуються для

демонстрації контролю над DID.

- assertionMethod - вказує, які ключі дозволено використовувати для випуску тверджень.
- keyAgreement - визначає ключі для встановлення безпечних каналів зв'язку.service - містить набір сервісних кінцевих точок (serviceEndpoint), пов'язаних з DID-суб'єктом.

Базовою характеристикою DID виступає його криптографічний зв'язок з публічним ключем суб'єкта. При створенні DID генерується пара ключів (приватний та публічний). Приватний ключ зберігається в захищеному середовищі (цифровий гаманець, апаратний модуль) і використовується для підпису даних. Публічний ключ включається до DID-документа та публікується в розподіленому реєстрі. Таким чином, будь-хто, хто має DID, може отримати відповідний DID-документ, дізнатися публічний ключ суб'єкта та перевірити цифровий підпис [20].

Нижче, на Рисунку 1.2, наведено приклад DID-документу, створеного під час реєстрації IoT-пристрою в тестовій мережі Polygon Amoy.


```
{
"@context": [ "https://www.w3.org/ns/did/v1"
],
"id": "did:polygon:testnet:0xB07598817884d6336e27508f7379Da8F4F3891A8", "verificationMethod": [
{
        "id": "did:polygon:testnet:0xB07598817884d6336e27508f7379Da8F4F3891A8#keys-1",
        "type": "Ed25519VerificationKey2020",
        "controller": "did:polygon:testnet:0xB07598817884d6336e27508f7379Da8F4F3891A8",
        "blockchainAccountId": "0xB07598817884d6336e27508f7379Da8F4F3891A8@polygon:testnet"
        }
  ],
  "authentication": [
        "did:polygon:testnet:0xB07598817884d6336e27508f7379Da8F4F3891A8#keys-1"
  ]
}
```


Рисунок 1.2 - Приклад DID-документу

У цьому документі:

- поле id містить DID пристрою;
- поле verificationMethod описує публічний ключ (тип Ed25519) та блокчейн-

пристрою;

- поле authentication вказує, який метод використовується для доведення контролю над DID.

DID охоплюють усі стадії існування ідентичності, забезпечуючи її повний цикл управління:

- Створення (Creation) - генерація пари ключів, формування DID, публікація DID-документа в розподіленому реєстрі.
- Оновлення (Update) - зміна DID-документа (наприклад, ротація ключів, оновлення сервісних точок) шляхом підписання оновленої версії старим ключем.
- Деактивація (Deactivation) - позначення DID як недійсного (наприклад, при виведенні пристрою з експлуатації) без можливості подальшого оновлення.

Усі операції життєвого циклу виконуються без участі центрального органу, що відповідає принципам децентралізації.

DID є фундаментальним будівельним блоком децентралізованих систем ідентифікації. Він забезпечує унікальну, стійку до втрати та криптографічно підтверджувану ідентифікацію осіб без участі центральних органів. DID-документ публікується в розподіленому реєстрі (блокчейні), що гарантує його незмінність та загальнодоступність.

1.6 Порівняльний аналіз DID методів для IoT

Для практичної реалізації децентралізованої системи управління ідентичністю необхідно обрати конкретний DID метод, який визначає, де і як зберігатимуться DID-документи та як виконуватиметься їх резолвінг. Спираючись на виконаний аналіз, було досліджено ключові методи DID, придатні для використання в IoT-середовищах [16]:

1. Метод did:key є самосертифікованим: DID безпосередньо представляє криптографічний публічний ключ і не потребує зовнішнього реєстру. Це найпростіший метод з нульовою вартістю та миттєвим підтвердженням.

Однак його ключове обмеження - неможливість оновлення DID-документа (наприклад, при ротації ключів) або його деактивації. Тому did:key придатний лише для статичних, одноразових ідентифікаторів, але не для довготривалого управління ідентичністю IoT-пристроїв [17].Метод did:ethr базується на смарт-контрактах у мережі Ethereum (L1 або L2). Блокчейни мають різні рівні (layers). L1 (Layer 1) - це базовий рівень, наприклад Ethereum або Bitcoin. Транзакції тут підтверджуються безпосередньо в основній мережі, але це повільно та дорого. L2 (Layer 2) - це додаткові протоколи (наприклад, Polygon, Arbitrum, Optimism), які працюють поверх L1. Вони обробляють транзакції швидше та з нижчими комісіями, а згодом із заданою періодичністю фіксують їх у головну мережу L1. DID методи можуть працювати як на L1, так і на L2. DID формується з Ethereum-адреси, а DID-документ зберігається в смарт-контракті. Переваги: високий рівень децентралізації, зріла екосистема інструментів, підтримка оновлення та деактивації. Основний недолік - необхідність сплати «газових» комісій за кожну операцію. Навіть у L2-мережах ці комісії є ненульовими, що для сценаріїв з частими оновленнями може бути економічно недоцільним [16].

2. Метод did:polygon (використаний у даній роботі) є адаптацією did:ethr для мережі Polygon (яка є L2-рішенням для Ethereum). DID формується за тим самим принципом: did:polygon:testnet:0x...

Переваги:

- Безкоштовна тестова мережа (Polygon Amoy Testnet) - існує можливість отримати експериментальні токени POL через сервіс Faucet без реальних витрат
- Сумісність з MetaMask - для підпису транзакцій використовується стандартний Ethereum-гаманець
- Підтримка смарт-контрактів - реєстрація DID через функцію createDID
- Низька вартість - навіть у основній мережі Polygon комісії значно нижчі,

ніж у Ethereum

Недоліки: як і для did:ethr, залишається потреба в газових комісіях (хоча в тестовій мережі вони компенсуються Faucet).

Альтернативні методи (did:indy, did:iota) існують, але мають обмеження. did:indy є частиною дозволеної мережі Hyperledger Indy, що знижує рівень децентралізації. did:iota використовує технологію Tangle з нульовими комісіями, але його екосистема менш зріла, а підтримка легковагових клієнтів обмежена [17]. В даної роботи ці методи не використовувались

Таблиця 1.4 - Зіставна характеристика DID-методів, придатних для застосування в IoT

Критерій	did:key	did:ethr	did:polygon	did:indy	did:iota
Вартість реєстрації	0	~5-15 USD (газ)	< 0.01 USD	0 (в закритих мережах)	0
Час підтвердження	Миттєво	~15-30 с	Газові токени (низькі)	~5-10 с	~5-15 с
Ротація ключів	Ні	Так	Так	Так	Так
Синхронна верифікація	Ні	Ні	Ні	Ні	Ні
Єдина точка відмови	Ні	Ні	Ні	Так (permissioned)	Ні

Проведений аналіз показує, що для практичного застосування в IoT найбільш

придатними є публічні методи з низькою вартістю (did:polygon, did:iota). Метод did:indy є дозволеним (permissioned), що суперечить принципам децентралізації. Метод did:key не підтримує ротацію ключів. Метод did:ethr на рівні L1 є економічно недоцільним. Керуючись результатами цього дослідження, для наступної прикладної реалізації обрано «did:polygon».

1.7 Ресурсні обмеження IoT-пристроїв як фактор архітектури

Успішна реалізація криптографічних механізмів на IoT-пристроях неможлива без врахування їхніх суворих апаратних обмежень. Для того, щоб обрати придатні алгоритми (цифрового підпису, шифрування, хешування), необхідно кількісно оцінити, які ресурси є критичними.

До основних обмежень, що впливають на вибір криптографії, належать:

- Об'єм оперативної пам'яті (RAM) - визначає, чи зможе пристрій виконати алгоритм без переповнення.
- Об'єм постійної пам'яті (Flash/ROM) - визначає, чи поміститься код алгоритму необхідні дані.
- Тактова частота процесора - впливає на час виконання криптографічних операцій.
- Енергоспоживання - критичне для пристроїв, що працюють від батарей.

З метою кількісної оцінки проаналізуємо три типових класи IoT пристроїв, які відрізняються за обсягом ресурсів:

Таблиця 1.5 - Порівняння класів IoT-пристроїв

Клас	RAM	Flash	Частота	Енерго-споживання	Типові представники
Class 1 (надзвичайно обмежені)	2-32 КБ	32-128 КБ	8-32 МГц	0.5-10 мА	8-бітні мікроконтролери (ATmega328P)

Class 2 (обмежені)	64-520 КБ	256 КБ - 4 МБ	80-240 МГц	20-150 мА	32-бітні мікроконтролери (ESP32,STM32)
Class 3 (умовно обмежені)	512+ МБ	4+ ГБ	1+ ГГц	200-500 мА	Одноплатні комп'ютери (Raspberry Pi)

Продовження табл. 1.5

Вплив обмежень на криптографічні операції:

1. Обчислювальна потужність. Генерація RSA-ключа довжиною 2048 біт на пристрої Class 1 з частотою 16 МГц займає до 30 секунд, що робить її неприйнятною для більшості застосувань. На пристроях Class 2, що працюють на тактовій частоті 240 МГц час виконання аналогічної операції скорочується до сотень мілісекунд, але все ще залишається значним. Легковагові алгоритми здатні виконуватись за одиниці мілісекунд [17].
2. Пам'ять. Перевірка ланцюжка сертифікатів X.509, типова для PKI, вимагає понад 10 КБ RAM, що перевищує можливості пристроїв Class 1. Легковагові алгоритми цифрового підпису потребують 2-4 КБ RAM, а симетричні алгоритми шифрування - менше 1 КБ RAM [1; 11; 17].Постійна пам'ять. Розмір прошивки для підтримки повного стеку TLS з сертифікатами X.509 може сягати 50-100 КБ, що має вирішальне значення для пристроїв Class 1 та Class 2. Реалізація легковагових алгоритмів займає 5-15 КБ ROM [1; 11].
3. Пропускна здатність мережі. Типовий X.509 сертифікат займає 1-2 КБ. У технологіях LoRaWAN або Sigfox максимальний розмір пакету становить 51-243 байти, що унеможливлює передачу сертифіката. Легковагові формати даних, такі як DID-документ, мають розмір 150-200 байт, що дозволяє передавати їх в одному пакеті.
4. Енергоспоживання. Передача 1 КБ даних через Wi-Fi на ESP32 в активному

режимі споживає близько 0.5 Дж. Для пристрою, що працює від батареї ємністю 1000 мА·год, це дозволяє виконати близько 7000 таких передач. Скорочення розміру даних у 5-10 разів збільшує кількість можливих передач пропорційно.

Традиційні криптографічні механізми, такі як RSA та X.509, є непридатними для пристроїв Class 1 та Class 2 через значні вимоги до пам'яті, обчислювальної потужності та пропускної здатності. Необхідно використовувати легковагові алгоритми, які забезпечують:

- малий розмір ключів (до 64 байт);
- малий розмір підписів (до 128 байт); низьке споживання RAM (до 2 КБ);
- високу швидкість виконання на 32-бітних процесорах (мілісекунди, а не секунди).

1.8 Легковагова криптографія: алгоритми та стандарти (NIST, ISO)

Для подолання ресурсних обмежень, описаних у підрозділі 1.7, необхідно використовувати легковагову криптографію - клас алгоритмів, спеціально оптимізованих для роботи в умовах обмеженої пам'яті, низької тактової частоти та мінімального енергоспоживання. Процесами нормування легковагових криптографічних рішень опікуються провідні міжнародні організації. Національний інститут стандартів і технологій США (NIST) розпочав процес стандартизації легковагової криптографії у 2015 році, а у 2025 році фіналізував стандарт ASCON. Міжнародна організація зі стандартизації (ISO) розробила стандарт ISO/IEC 29192, що визначає перелік легковагових алгоритмів для різноманітних сфер діяльності, та ISO/IEC 29167, зосереджений на криптографічних технологіях для RFID-пристроїв [23], [24].

Для використання в IoT-системах розглядаються різні класи алгоритмів: блокові шифри (PRESENT, SPECK, SIMON), потокові шифри (ChaCha20, ASCON) та алгоритми цифрового підпису (Ed25519, ECDSA). Для кількісного порівняння цих алгоритмів за критеріями, важливими для обмежених пристроїв, нижче

наведено порівняльну таблицю.

Таблиця 1.6 - Порівняння легковагових криптографічних алгоритмів для IoT

Тип алгоритму	Алгоритм	Розмір ключа (байт)	Розмір підпису/блоку (байт)	RAM (КБ)	ROM (байт)	Час виконання на 32-бітному CPU (мс)
Блоковий шифр	SPECK - 64/128	16	8 (блок)	~0.1	~500	< 0.1 мс
Блоковий шифр	SIMON-64/128	16	8 (блок)	~0.1	~500	< 0.1 мс
Потоковий шифр	ChaCha20	32	-	~1	~3000	~0.24 на 1 КБ
Потоковий шифр	ASCON-128	16	-	~0.5	~2000	~0.5 на 1 КБ
Цифровий підпис	Ed25519	32	64	~2	~10000	~2.5 (підпис), ~60 (верифікація)
Цифровий підпис	ECDSA (secp256k1)	32	64	~3	~15000	~4 (підпис), ~70 (верифікація)

На основі проведеного аналізу для програмної реалізації обираються наступні алгоритми.

Для цифрового підпису (автентифікація) обрано Ed25519. ECDSA не обрано через вище споживання RAM (3 КБ проти 2 КБ) та ризик слабких випадкових

чисел, необхідних генерації nonce.

Для симетричного шифрування (конфіденційність) обрано ChaCha20. SPECK та SIMON не забезпечують вбудованої автентифікації даних, тому їх використання потребує додаткового режиму (наприклад, GCM), що збільшує обчислювальне навантаження. ASCON має нижче споживання RAM, але його продуктивність є нижчою згідно з експериментальними даними NIST [25] (близько 0.5 мс на 1 КБ даних проти 0.24 мс у ChaCha20 [1]). Тому ChaCha20 є кращим вибором для систем, де важлива швидкість.

Для хешування обрано SHA-256. Цей алгоритм є загальноприйнятим стандартом, має апаратне прискорення на ESP32 та забезпечує достатній рівень безпеки (256 біт) [25].

1.9 Функціональні та нефункціональні вимоги до системи

На основі раніше проведеного аналізу, сформульовано наступні вимоги до системи.

Таблиця 1.7 - Функціональні вимоги до системи

ID	Вимога	Опис
FR1	Реєстрація пристрою	Система повинна забезпечувати створення унікального DID для кожного IoT-пристрою та його реєстрацію в блокчейні.
FR2	Генерація ключів	Пристрій повинен генерувати криптографічну пару ключів Ed25519.
FR3	Підпис даних	Пристрій повинен мати можливість підписувати телеметричні дані своїм приватним ключем.
FR4	Шифрування даних	Пристрій повинен мати можливість шифрувати телеметричні дані перед збереженням або передачею.

FR5	Публікація DID-документа	Система повинна публікувати DID-документ у децентралізованому сховищі для загальнодоступної верифікації.
FR6	Верифікація підпису	Будь-яка сторона повинна мати можливість перевірити цифровий підпис даних, використовуючи публічний ключ з DID-документа.
FR7	Збереження локального профілю	Система повинна зберігати локальний JSON-файл з усіма даними пристрою (ключі, DID, зашифровані показники, підписи).

Продовження табл. 1.7

Таблиця 1.8 – Нефункціональні вимоги до системи:

ID	Вимога	Вимога	Обґрунтування
FR1	Час генерації ключів	≤ 100 мс	Типовий час для Ed25519 на сучасних процесорах [5]
FR2	Час підпи у даних	≤ 10 мс	Забезпечення швидкої обробки телеметрії
FR3	Час шифрування (1 КБ)	≤ 1 мс	ChaCha20 є високопродуктивним [1]
FR4	Розмір DID-документа	≤ 500 байт	Можливість передачі в обмежених мережах (LoRaWAN)
FR5	Вартість реєстрації	<0.01 USD (тестова мережа - 0)	Polygon Amoy Testnet надає безкоштовні токени
FR6	Стійкість підробки	Неможливість підробити підпис без приватного ключа	Забезпечується криптостійкістю Ed25519

1.10 Нормативно-правова база

При розробці системи враховано вимоги наступних нормативних документів України у сфері кібербезпеки та захисту інформації:

- Закон України «Про захист інформації в інформаційно-комунікаційних системахвизначає правові основи захисту інформації в інформаційно-комунікаційних системах.
- Закон України «Про основні засади забезпечення кібербезпеки України» – формує правове та організаційне підґрунтя для гарантування кібербезпеки.
- ДСТУ ISO/IEC 27001:2015 – вимоги до системи управління інформаційною безпекою (використано при формуванні вимог до системи FR1-FR4).
- НД ТЗІ 2.5-004-99 (Криптографічний захист інформації) – визначено вимоги до використання криптографічних алгоритмів (Ed25519, ChaCha20).
- Стандарт W3C Decentralized Identifiers (DID) v1.0 – базовий стандарт, на якому побудовано архітектуру системи.

Створена система узгоджується з фундаментальними засадами охорони інформації, визначеним зазначеними нормативними документами, зокрема забезпечує конфіденційність, цілісність та доступність інформації, а також автентифікацію джерела даних за допомогою цифрового підпису Ed25519.

Висновки до розділу 1

Теоретичне дослідження, виконане в першому розділі, дає змогу зробити наступні висновки, на основі яких буде виконано проєктування архітектури системи.

По-перше, виявлені фундаментальні недоліки, притаманні централізованим і федеративним моделям ідентифікації, зокрема PKI, OAuth2, SAML та Shibboleth. Ці моделі мають єдину точку відмови у вигляді центрів сертифікації та провайдерів ідентичності, страждають від проблем масштабування, високих

затримок та складності управління сертифікатами. Отже, для IoT-середовища необхідна децентралізована архітектура.

По-друге, парадигма самостійно керованої ідентичності (SSI) та стандарти W3C DID формують теоретичний фундамент для децентралізованого управління ідентичностями. DID-документ дозволяє публікувати криптографічні ключі пристрою в розподіленому реєстрі, що забезпечує їх незмінність та загальнодоступність.

По-третє, порівняльний аналіз DID методів показав, що для прикладного впровадження найбільш доцільним є метод did:polygon. Його переваги - безкоштовна (Polygon Amoy Testnet), сумісність з MetaMask, підтримка смарт-контрактів та низька вартість транзакцій - є вирішальними для створення демонстраційного прототипу.

По-четверте, аналіз ресурсних обмежень IoT-пристроїв та легковагової криптографії дозволив визначити криптографічний стек для програмної реалізації. Для автентифікації (цифрового підпису) обрано алгоритм Ed25519, який має малий розмір ключа (32 байти) та підпису (64 байти) та є детерміністичним. Для забезпечення конфіденційності (шифрування) обрано алгоритм ChaCha20, який демонструє високу продуктивність на 32-бітних процесорах та стійкість до атак побічними каналами. Для забезпечення цілісності (хешування) обрано алгоритм SHA-256, який є загальноприйнятим стандартом та забезпечує достатній рівень безпеки.

На основі цих висновків розроблено архітектуру децентралізованої системи управління ідентичністю для IoT-пристроїв, яка реалізує обрані технології: DID метод did:polygon та криптографічний стек Ed25519, ChaCha20, SHA-256. Будуть визначені модулі системи, протоколи взаємодії між пристроєм, а також створено схеми організації даних для процедур реєстрації, підпису та верифікації телеметричних даних.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ДЕЦЕНТРАЛІЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ІДЕНТИЧНІСТЮ ДЛЯ IOT-ПРИСТРОЇВ

2.1 Призначення та бізнес-логіка системи

Розроблена система призначена для децентралізованої реєстрації IoT-пристроїв та криптографічного захисту їхніх телеметричних даних. Система забезпечує три ключові функції безпеки:

- ідентифікацію - кожен пристрій отримує унікальний децентралізований ідентифікатор (DID);
- автентифікацію - цифровий підпис Ed25519 доводить, що дані надійшли саме від зареєстрованого пристрою;
- конфіденційність - симетричне шифрування ChaCha20 захищає дані від несанкціонованого перегляду.

Система орієнтована на сценарії використання, де критично справжність пристрою та цілісність його даних без залучення централізованих центрів сертифікації. Прикладом такого сценарію є безпека розумного будинку: датчики контролю дверей, вікон та детектори руху повинні надсилати справжні дані, а власник повинен мати змогу перевірити, що ці дані не були підроблені.

Специфіка прототипу. У розробленому прототипі IoT-пристрій симулюється програмно на Node.js. Фізичне обладнання (ESP32, датчики, камери) не використовується, оскільки метою роботи є демонстрація децентралізованої ідентифікації, криптографічного захисту даних та публічної верифікації, а не робота з конкретними фізичними датчиками. Всі функції пристрою (генерація ключів, збір телеметрії, шифрування, підписання, взаємодія з IPFS та блокчейном) реалізовані програмно в середовищі Node.js.

Бізнес-логіка системи включає наступний сценарій:

1. Реєстрація нового пристрою - генерація пари ключів Ed25519, створення DID, публікація DID-документа в IPFS, реєстрація DID у смарт-контракті Polygon, збереження локального профілю пристрою.

2. Шифрування та підписання телеметричних даних - пристрій виконує шифрування вимірюваних величин із застосуванням ChaCha20, після чого підписує їх своїм приватним ключем Ed25519.
3. Верифікація даних - отримувач (Verifier) завантажує DID-документ з IPFS, отримує публічний ключ, розшифровує дані та перевіряє цифровий підпис.

Гарантії безпеки, що забезпечуються системою:

- Автентичність - підпис Ed25519 доводить, що дані надіслано саме власником відповідного приватного ключа.
- Цілісність - будь-яка зміна даних призводить до недійсності підпису.
- Незмінність - інформація про реєстрацію DID у розподіленому реєстрі є загальнодоступною, незмінним доказом існування пристрою.
- Конфіденційність - шифрування ChaCha20 забезпечує захист від несанкціонованого перегляду даних.

2.2 Архітектура системи (компоненти та взаємодія)

На основі сформульованих у підрозділі 1.9 функціональних та нефункціональних вимог розроблено архітектуру децентралізованої системи управління цифровою для IoT-пристроїв (DID for IoT). Система реалізує три ключові функції безпеки: ідентифікацію (кожен пристрій отримує унікальний DID), автентифікацію (цифровий підпис Ed25519 доводить справжність даних) та конфіденційність (симетричне шифрування).

Архітектура системи складається з трьох основних компонентів, кожний із яких реалізує чітко окреслену функцію у життєвому циклі DID (рис. 2.1).

1. IoT-пристрій (клієнт реєстрації).

Програмний модуль реалізовує логіку IoT-пристрою та виконує наступні функції:

- генерація криптографічної пари ключів;
- створення DID-документа у форматі JSON-LD відповідно до стандарту W3C;
- збір телеметричних даних;

- шифрування даних (забезпечення конфіденційності);
- підписання даних за допомогою приватного ключа пристрою;
- реєстрація DID у смарт-контракті блокчейну Polygon;
- публікація DID-документа в IPFS;
- збереження всіх даних у локальний JSON-файл.

2. Блокчейн Polygon (Amoy Testnet). Блокчейн виконує роль децентралізованого реєстру DID (Decentralized Identifier Registry). Смарт-контракт з визначеною адресою реалізує функцію createDID, яка реєструє зв'язок між адресою власника та DID-рядком, створює подію DIDRegistered з метою організації аудиторського контролю та повертає ідентифікатор реєстрації. Кожна реєстрація створює транзакцію, хеш якої (registrationTxHash) є незмінним публічним доказом існування пристрою на момент реєстрації. Транзакція може бути перевірена через блокчейн-експлорер PolygonScan.
3. IPFS (Pinata). IPFS використовується як децентралізоване сховище DID-документів. Процес публікації включає: завантаження DID-документа (JSON) в IPFS через API Pinata з використанням автентифікації за допомогою API-ключа та секретного ключа (Pinata API Key та Secret Key); отримання унікального CID (Content Identifier) - криптографічного хеша вмісту документа; збереження CID у локальному профілі пристрою (поле didDocumentIpfsHash). Будь-яка сторона (верифікатор) може DID-документ за CID через публічний шлюз. Оскільки CID є хешем вмісту, підробка документа неможлива - будь-яка зміна даних призводить до зміни CID.

У розробленому прототипі всі компоненти, крім блокчейну та IPFS, симулюються програмно на Node.js. Фізичні IoT-пристрої не використовуються. На рисунку 2.1 наведена компонентна діаграма архітектури системи, яка ілюструє описану взаємодію між компонентами.

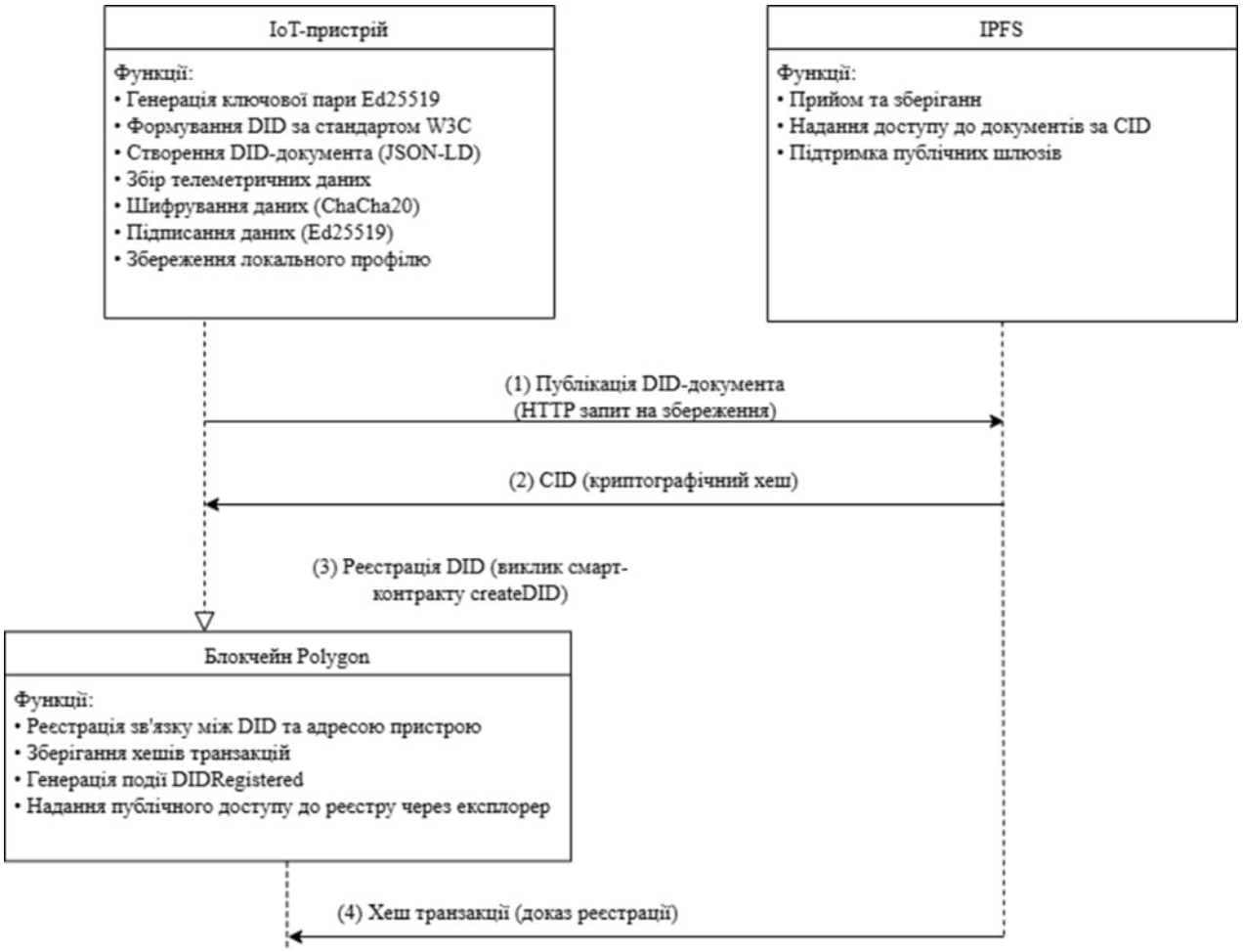


Рисунок 2.1 - Компонентна діаграма архітектури системи

З метою ґрунтовного усвідомлення просторового розташування складових системи в середовищі виконання розроблено діаграму розгортання (deployment diagram), наведену на рисунку 2.2. Діаграма показує, на яких фізичних вузлах (серверах, пристроях, зовнішніх сервісах) розгортаються програмні компоненти системи.

Система розгортається на чотирьох типах фізичних вузлів:

1. Веб-браузер користувача (User Browser) – запускається на комп'ютері або мобільному пристрої користувача. Виконує клієнтський JavaScript-код (файл index.html), забезпечує відображення веб-інтерфейсу, зчитування даних з форми, надсилання HTTP-запитів до сервера та відображення результатів реєстрації (DID, TxHash, CID).

2. Сервер реєстрації (Registration Server) – запускається на виділеному сервері або локальній машині розробника (операційна система Windows/Linux/macOS, середовище Node.js). Виконує основну бізнес-логіку: функцію registerDevice() (генерація ключів, створення DID, шифрування, підписання), HTTP-сервер на базі Express для обробки API-запитів, а також зберігає локальні JSON-файли профілів пристроїв у файловій системі.
3. Блокчейн Polygon Amoy Testnet (зовнішній розподілений вузол) – фізично перебуває в межах глобальної мережі, утвореної вузлами блокчейну Polygon. Виконує смарт-контракт DIDRegistry з функцією createDID, зберігає реєстр відповідностей «власник → DID», забезпечує незмінність та публічну верифікацію транзакцій через PolygonScan.
4. IPFS + Pinata Gateway (зовнішній сервіс) – фізично розміщений на серверах компанії Pinata (хмарне сховище). Надає API для завантаження файлів (pinFileToIPFS) та публічний шлюз для отримання файлів за CID (https://gateway.pinata.cloud/ipfs/{CID}). Забезпечує децентралізоване зберігання DID-документів.

У розробленому прототипі сервер реєстрації та веб-браузер можуть запускатися на одному фізичному комп'ютері (наприклад, localhost:3000). Це спрощує відлагодження та тестування. У промисловому розгортанні сервер доцільно виносити на окремий хост з публічною IP-адресою для доступу з будь-яких клієнтських пристроїв.

Діаграма розгортання на рисунку 2.2 візуалізує описані фізичні вузли, з'єднання між ними та протоколи взаємодії.

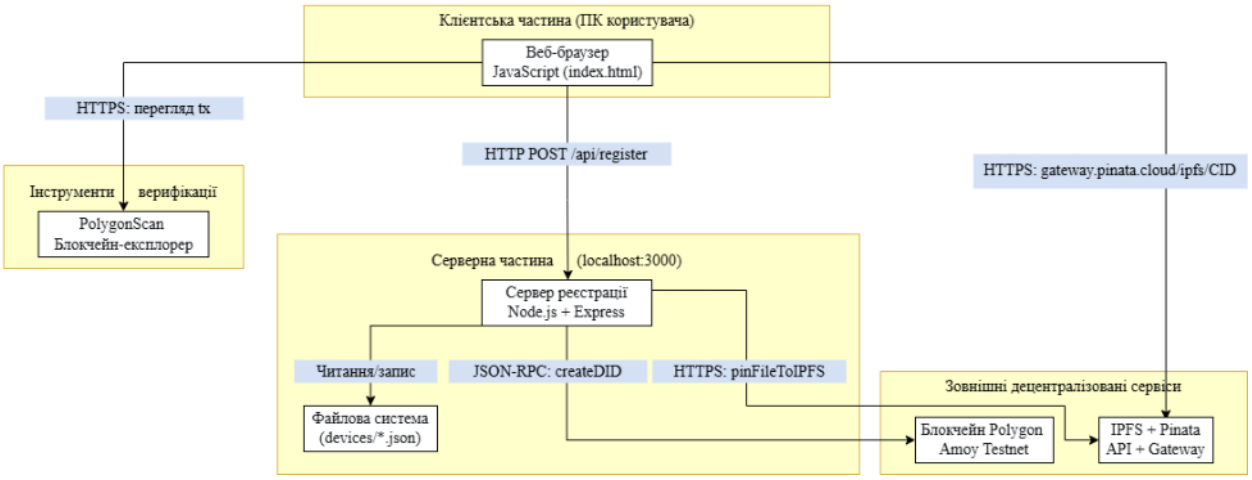


Рисунок 2.2 - Діаграма розгортання системи

Розроблена архітектура забезпечує наступні гарантії безпеки. Автентичність забезпечується цифровим підписом Ed25519. Цілісність гарантується тим, що будь-яка зміна підписаного повідомлення робить підпис недійсним - це перевіряється верифікацією підпису з використанням публічного ключа. Незмінність реєстрації забезпечується транзакцією в блокчейні, хеш якої (registrationTxHash) можна перевірити в PolygonScan. Незмінність DID-документа забезпечується тим, що CID (криптографічний хеш вмісту) унеможливлює підробку документа - будь-яка зміна даних призводить до зміни CID. Конфіденційність забезпечується шифруванням.

Запропонована архітектура втілює основоположні засади децентралізованої ідентичності (Self-Sovereign Identity), розглянуті в підрозділі 1.4.

Принцип контролю реалізується тим, що IoT-пристрій генерує власну пару ключів та створює DID без участі центрального провайдера. Приватний ключ зберігається локально та ніколи не передається третім сторонам.

Принцип портативності реалізується тим, що DID не прив'язаний до жодного виробника, платформи або сервісу. Ідентичність пристрою може бути використана в будь-якій системі, що підтримує стандарт W3C DID.

Принцип сумісності реалізується тим, що DID-документ відповідає специфікації W3C, що забезпечує сумісність з іншими DID-сумісними системами та

інструментами.

Принцип персистентності реалізується тим, що DID є довготривалим ідентифікатором, який залишається незмінним протягом усього життєвого циклу пристрою (навіть при ротації ключів, яка створює новий DID-документ, але старий DID може зберігатися для історії).

2.3 Протокол реєстрації IoT-пристрою

Протокол реєстрації визначає логічну послідовність дій, необхідних для присвоєння IoT-пристрою децентралізованого ідентифікатора (DID) та публікації доказів його існування в розподіленому реєстрі. Протокол складається з шести етапів, послідовність яких наведена на рисунку 2.2.

Етап 1. Генерація ключової пари пристрою. На цьому етапі IoT-пристрій генерує власну асиметричну криптографічну пару ключів. Приватний ключ залишається на пристрої та ніколи не передається в мережу. призначений для публікації та подальшої верифікації. Обраний алгоритм - Ed25519, який забезпечує малий розмір ключа (32 байти), високу швидкість генерації на обмежених пристроях та детерміністичність.

Етап 2. Формування DID та DID-документа. На основі публічної адреси, отриманої з публічного ключа, пристрій формує унікальний DID за шаблоном, визначеним обраним DID методом. DID-документ формується у

форматі JSON-LD відповідно до специфікації W3C й включає такі невід'ємні складники: ідентифікатор контексту, власне DID пристрою, опис методу верифікації (тип ключа та посилання на блокчейн-аккаунт), а також перелік методів автентифікації.

Етап 3. Публікація DID-документа в децентралізованому сховищі. DID-документ публікується в децентралізованій файловій системі (IPFS). Це забезпечує: незмінність вмісту (кожен документ отримує унікальний криптографічний хеш - CID); доступність через публічні шлюзи; відсутність єдиного контролера даних. Результатом публікації є CID, який зберігається в локальному профілі пристрою

для подальшого використання верифікаторами.

Етап 4. Захист телеметричних даних. Пристрій збирає актуальні телеметричні дані (у наведеному сценарії - статус безпеки: стан дверей, вікон, наявність руху). З метою гарантування конфіденційності інформація захищається за допомогою симетричного криптоалгоритму ChaCha20-Poly1305, який забезпечує високу швидкість на 32-бітних процесорах та вбудовану автентифікацію. Для забезпечення автентичності та цілісності оригінальні (незашифровані) дані підписуються приватним ключем пристрою. Це дозволяє будь-якому верифікатору переконатись, що дані дійсно надійшли від зареєстрованого пристрою та не були змінені під час передачі.

Етап 5. Реєстрація DID у розподіленому реєстрі. Власник пристрою (його гаманець) ініціює транзакцію в блокчейні, яка реєструє зв'язок між DID та адресою пристрою. Транзакція включає виклик функції смарт-контракту createDID з параметрами: адреса пристрою та рядок DID. Транзакція підписується приватним ключем власника, який, крім того, бере на себе зобов'язання щодо сплати газових зборів. Результатом реєстрації є хеш транзакції, який слугує незмінним публічним доказом існування пристрою на момент реєстрації та може бути перевірений через будь-який блокчейн-експлорер.

Етап 6. Збереження локального профілю пристрою. Після успішного виконання всіх попередніх етапів пристрій зберігає локально всі необхідні дані для подальшої роботи: приватний ключ пристрою, DID, хеш транзакції реєстрації (доказ в блокчейні), CID DID-документа (доказ в IPFS), закодована телеметрична інформація разом із вектором ініціалізації (IV) та тегом автентифікації, а також цифровий підпис оригінальних даних. Цей локальний профіль дозволяє пристрою відновити свій стан після перезавантаження та надавати верифікаторам усі необхідні докази автентичності.

На рисунку 2.3 наведено діаграму послідовності (sequence diagram) протоколу реєстрації, яка ілюструє порядок передачі повідомлень між компонентами

системи.

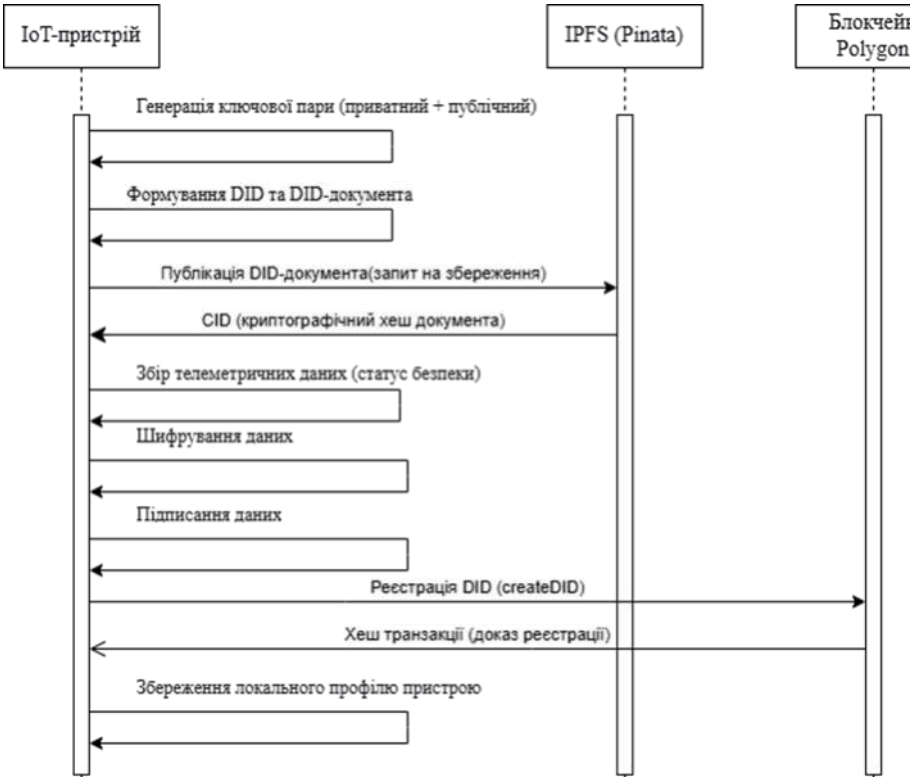


Рисунок 2.3 - Діаграма послідовності реєстрації IoT-пристрою

2.4 Схема публікації DID-документа в децентралізованому сховищі (IPFS)

Після створення DID-документа він має бути доступним для будь-якої сторони, яка хоче верифікувати пристрій. Для цього документ публікується в децентралізованій файловій системі, яка забезпечує:

- Незмінність вмісту - кожен документ отримує унікальний криптографічний хеш (CID); будь-яка зміна документа призводить до зміни хеша.
- Доступність - документ може бути отриманий з будь-якої точки світу через публічні шлюзи.
- Децентралізацію – відсутній єдиний контролер даних.

Процес публікації DID-документа складається з наступних етапів, зображених на рисунку 2.4.

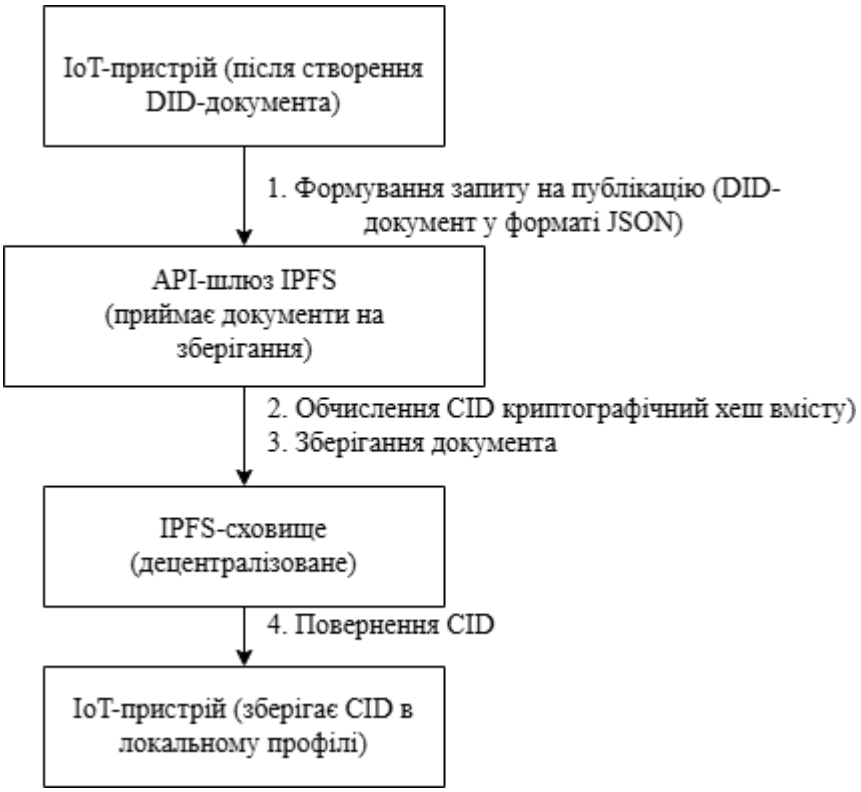


Рисунок 2.4 - Схема публікації DID-документа в IPFS

Кожен DID-документ отримує унікальний ідентифікатор - CID (Content Identifier). CID є криптографічним хешем вмісту документа, що забезпечує:

- Цілісність - при зміні документа змінюється CID, що робить підробку неможливою.
- Постійність - CID не прив'язаний до конкретної локації розміщення документа.
- Відсутність необхідності в центральному реєстрі - CID може бути отриманий будь- без звернення до центрального сервера.

Будь-яка сторона (верифікатор) може отримати DID-документ за CID двома способами:

- Через публічний шлюз - HTTP-запит до шлюзу за адресою https://шлюз/ipfs/{CID}.
- Через локальний IPFS-вузол - прямий запит до власного вузла в мережі IPFS.

Після отримання документа верифікатор вилучає з нього публічний ключ пристрою для подальшої перевірки підпису.

2.5 Схема реєстрації DID у розподіленому реєстрі (блокчейн Polygon)

Реєстрація DID у блокчейні створює незмінний, публічний доказ існування пристрою на момент реєстрації. Це усуває потребу в централізованому центрі сертифікації (CA) а також виключає можливість фальсифікації чи блокування ідентифікатора з боку третіх сторін.

Процес реєстрації DID у блокчейні складається з етапів, представлених на рисунку 2.5.

Рисунок 2.5 - Схема реєстрації DID у блокчейні Polygon

Смарт-контракт реалізує функцію createDID з наступними параметрами.

Таблиця 2.1 - Параметри функції createDID:

Параметр	Тип	Опис
_owner	address	Адреса власника пристрою (гаманця)
_didString	string	DID-рядок пристрою

Логіка виконання функції:

1. Перевіряє, чи DID ще не зареєстрований.
2. Зберігає зв'язок owner -> did у внутрішньому реєстрі.
3. Генерує подію DIDRegistered для аудиту.
4. Повертає ідентифікатор реєстрації.

Результатом реєстрації є хеш транзакції (transaction hash), який:

- Є унікальним ідентифікатором запису в блокчейні.
- Може бути публічно перевірений через будь-який блокчейн-експлорер.
- Слугує незмінним доказом того, що пристрій був зареєстрований у вказаний час.

2.6 Протокол верифікації даних

Протокол верифікації дозволяє будь-якій стороні (верифікатору) перевірити автентичність та цілісність даних, отриманих від IoT-пристрою, без звернення до центрального сервера. Верифікація виконується з використанням лише:

- DID пристрою (отриманого з пакета даних)
- Публічного ключа (з DID-документа в IPFS)
- Цифрового підпису (з пакета даних)

Процес верифікації складається з наступних етапів, зображених на рисунку 2.6.

Рисунок 2.6 - Схема протоколу верифікації даних

Алгоритм верифікації виконує наступні кроки:

Вхідні дані:

- DID пристрою
- Оригінальні дані (або зашифровані)
- Цифровий підпис Вихідні дані:
- VALID або INVALID

Кроки:

- Отримати CID DID-документа (з локального профілю або з блокчейну).
- Завантажити DID-документ з IPFS за CID.
- Вилучити публічний ключ з поля verificationMethod.
- Обчислити хеш оригінальних даних.
- Перевірити підпис за допомогою публічного ключа.
- Повернути результат.

Розроблений протокол верифікації забезпечує наступні гарантії безпеки, наведені в таблиці 2.3.

Таблиця 2.3 - Гарантії безпеки протоколу верифікації

Гарантія	Як забезпечується
Автентичність	Дійсний підпис доводить, що дані створені саме власником приватного ключа
Цілісність	Будь-яка зміна даних після підписання робить підпис недійсним
Децентралізація	Верифікатор не звертається до центрального сервера - достатньо IPFS та блокчейну
Стійкість до підробки	Неможливо підробити підпис Ed25519 без приватного ключа

Висновки до розділу 2

У другому розділі кваліфікаційної роботи виконано проєктування децентралізованої системи управління цифровою ідентичністю для IoT-пристроїв на основі технології децентралізованих ідентифікаторів (DID).

Визначено призначення та бізнес-логіку системи. Система забезпечує три ключові функції безпеки: ідентифікацію (кожен пристрій отримує унікальний DID), автентифікацію (цифровий підпис Ed25519 доводить справжність даних) та конфіденційність (симетричне шифрування ChaCha20). Описано сценарії використання: реєстрація нового пристрою, шифрування та підписання телеметричних даних, верифікація даних отримувачем.

1. Розроблено архітектуру системи. Система складається з трьох основних компонентів: IoT-пристрій (клієнт реєстрації), блокчейн Polygon Amoy Testnet (децентралізований реєстр DID) та IPFS через сервіс Pinata (децентралізоване сховище DID-документів). Визначено функції кожного компонента та їхню взаємодію. Розроблена архітектура забезпечує автентичність, цілісність, незмінність реєстрації та конфіденційність даних, а також реалізує ключові принципи самостійно керованої ідентичності (SSI): контроль, портативність, сумісність та персистентність.
2. Сформульовано протокол реєстрації IoT-пристрою. Протокол складається з шести послідовних етапів: генерація ключової пари Ed25519; формування DID та DID-документа за стандартом W3C; публікація DID-документа в IPFS (отримання CID); шифрування телеметричних даних (ChaCha20-Poly1305) та їх підписання (Ed25519); реєстрація DID у смарт-контракті блокчейну Polygon (отримання хешу транзакції); збереження локального профілю пристрою у JSON-файл.
3. Розроблено схему публікації DID-документа в IPFS. Визначено, що кожному документу присвоюється неповторний криптографічний хеш (CID), який забезпечує незмінність вмісту та доступність через публічні

шлюзи. Процедура підтвердження справжності для доступу до API Pinata здійснюється за допомогою API-ключа та секретного ключа.

4. Розроблено схему реєстрації DID у блокчейні Polygon. Смарт-контракт реалізує функцію createDID, що фіксує взаємозв'язок між адресою власника та DID-рядком, генерує подію DIDRegistered для аудиту та повертає хеш транзакції. Хеш транзакції є незмінним публічним доказом існування пристрою.
5. Розроблено протокол верифікації даних. Протокол дозволяє будь-якій стороні перевірити автентичність та цілісність даних без звернення до центрального сервера, використовуючи лише DID пристрою, публічний ключ з DID-документа в IPFS та цифровий підпис. Процес перевірки автентичності містить шість послідовних етапів й повертає результат VALID або INVALID.
6. Визначено специфіку прототипу. У розробленому прототипі IoT-пристрій симулюється програмно на Node.js без використання фізичного обладнання (ESP32, датчики, камери). Всі компоненти, крім блокчейну Polygon та IPFS (які є реальними зовнішніми сервісами), реалізовані програмно. Такий підхід є допустимим для демонстраційного прототипу (Proof of Concept), тому що головним завданням виступає підтвердження функціональності механізму децентралізованої ідентифікації.

Таким чином, у другому розділі створено повноцінний проєкт децентралізованої системи управління ідентичністю, що охоплює всі функціональні та нефункціональні критерії, сформульовані в розділі 1, та є основою для програмної реалізації прототипу, опис якої наведено в третьому розділі.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ТЕСТУВАННЯ

3.1 Інструменти реалізації (програмні бібліотеки)

З метою розробки програмного прототипу децентралізованої платформи управління цифровою ідентичністю для IoT-пристроїв (DID for IoT) обрано наступний стек технологій та бібліотек.

Прототип реалізовано на платформі Node.js. Вибір обумовлений наступними факторами:

- Крос-платформеність - код може виконуватись на будь-якій операційній системі (Windows, Linux, macOS).
- Наявність вбудованого модуля crypto для виконання криптографічних операцій (хешування SHA-256, симетричне шифрування ChaCha20-Poly1305).
- Розвинене коло зовнішніх програмних бібліотек, призначених для комунікації з блокчейном (ethers.js), IPFS (Pinata), створення веб-сервера (Express) та обробки HTTP-запитів (axios).
- Можливість швидкого прототипування та відлагодження.

У прототипі використано програмні бібліотеки, продемонстровані в Таблиці 3.1.

Таблиця 3.1 - Програмні бібліотеки, використані в прототипі

Бібліотека	Версія	Призначення
ethers	6.13.0	Взаємодія з блокчейном Polygon Amoy Testnet: створення гаманця (Wallet), підписання повідомлень, виклик функцій смарт-контракту, отримання хешу транзакції.

crypto	вбудований модуль Node.js	Виконання криптографічних операцій: генерація випадкових байтів (для IV та ключів), симетричне шифрування та дешифрування (ChaCha20-Poly1305), хешування (SHA-256).
axios	1.7.2	Виконання HTTP-запитів до API Pinata для завантаження DID-документа в IPFS та отримання CID.
form-data	4.0.0	Формування multipart/form-data запитів для завантаження файлів в IPFS через API Pinata.
express	4.19.2	Створення веб-сервера (DID-резолвер) з REST API для реєстрації пристроїв, резолвінгу DID-документів, ротації ключів та деактивації.
fs	Вбудований модуль Node.js	Читання та запис локальних JSON-файлів (профілі пристроїв, база даних DID-документів).
dotenv	16.4.5	Завантаження конфігураційних змінних з файлу .env (PRIVATE_KEY, PINATA_API_KEY, PINATA_SECRET_KEY, RPC_URL, CONTRACT_ADDRESS)

Продовження табл. 3.1

Для роботи прототипу використовуються наступні зовнішні сервіси, наведені в Таблиці 3.2.

Таблиця 3.2 - Зовнішні сервіси, використані в прототипі

Сервіс	Призначення

Pinata (IPFS)	Децентралізоване зберігання DID-документів. Автентифікація виконується за допомогою API-ключа та секретного ключа. Після завантаження файлу сервіс повертає унікальний CID (криптографічний хеш вмісту).
Polygon Amoy Testnet	Тестова мережа блокчейну Polygon для реєстрації DID через смарт-контракт. Транзакції підписуються приватним ключем власника, зареєстрованим у MetaMask. Тестові токени POL отримуються через Faucet.
PolygonScan	Блокчейн-експлорер для публічної верифікації транзакцій реєстрації (перегляд хешу транзакції, статусу, блоку, часу).

Продовження табл. 3.2

Створення прототипу здійснювалося в середовищі Visual Studio Code з використанням розширень для JavaScript/Node.js та GitHub Copilot для прискорення написання коду. Управління версіями програмного коду виконувалося із застосуванням системи контролю версій Git.

Проект має наступну файлову структуру:

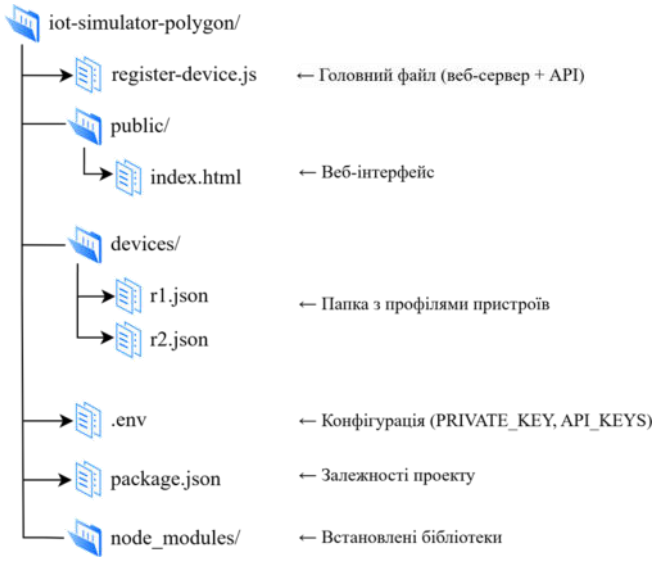


Рисунок 3.1 - Файлова структура проекту

3.2 Програмна реалізація реєстрації пристрою

Реєстрація нового IoT-пристрою є ключовою функцією прототипу. Вона реалізована у вигляді веб-інтерфейсу та API-ендпоінту /api/register, який викликає основну функцію registerDevice().

Процес реєстрації складається з наступних етапів, які відповідають протоколу, описаному в підрозділі 2.3:

1. Користувач заповнює веб-форму (назва пристрою + три параметри безпеки).
2. Браузер надсилає POST-запит на сервер (/api/register).
3. Сервер виконує функцію registerDevice():
    - Генерація ключової пари Ed25519.
    - Формування DID та DID-документа.
    - Публікація DID-документа в IPFS (отримання CID).
    - Шифрування телеметричних даних (ChaCha20-Poly1305).
    - Підписання даних (Ed25519).
    - Реєстрація DID у смарт-контракті Polygon (отримання хешу транзакції).
    - Збереження локального JSON-профілю.
4. Сервер повертає відповідь з DID, хешем транзакції та CID.
5. Веб-інтерфейс оновлює таблицю пристроїв.

Веб-інтерфейс реалізовано у файлі public/index.html. Він містить:

- Форму для введення назви пристрою.
- Три чекбокси для вибору стану безпеки (двері закриті?, вікна закриті?, рух виявлено?).
- Кнопку для запуску реєстрації.
- Таблицю для відображення списку зареєстрованих пристроїв.

На рисунку 3.2 наведено зовнішній вигляд веб-інтерфейсу.

Рисунок 3.2 - Головне вікно веб-інтерфейсу системи керування DID

Для кожного зареєстрованого пристрою в таблиці передбачено наступні кнопки дій:

- View - відображає повний JSON-профіль пристрою у нижній панелі.
- Verify - здійснює контроль автентичності цифрового підпису пристрою й демонструє результат верифікації.
- Copy TxHash - копіює хеш транзакції реєстрації в буфер обміну для подальшої перевірки в блокчейн-експлорері PolygonScan.
- Measurements - завантажує збережені виміри продуктивності пристрою та відображає таблицю з результатами (час генерації ключів, підписання, шифрування, розмір DID-документа, верифікація) з індикацією статусу відповідності вимогам FR1-FR4.
- Delete – видаляє пристрій із системи після підтвердження користувачем.

Для забезпечення програмної взаємодії між веб-інтерфейсом, описаним вище, та серверною логікою реєстрації IoT-пристроїв, розроблено REST API, яке надає набір ендпоінтів (кінцевих точок доступу). Кожна кінцева точка доступу (ендпоінт) забезпечує реалізацію певної дії, передбаченої вимогами до системи і представлений у таблиці 3.3.

Таблиця 3.3 - Основні API-ендпоінти системи

Ендпоінт	Метод	Призначення	Відповідна вимога

	HTTP		
/api/register	POST	Реєстрація нового IoT-пристрою	FR1, FR5
/api/devices	GET	Отримання списку всіх зареєстрованих пристроїв	FR6
/api/devices/:filename	GET	Отримання повного JSON-профілю конкретного пристрою	FR7
/api/devices/:filename	DELETE	Видалення пристрою з системи	Управління життєвим циклом
/api/verify	POST	Публічна верифікація цифрового підпису пристрою	FR6
/api/resolve/:ipfsHash	GET	Отримання DID-документа з IPFS за CID	FR5, FR6
/api/health	GET	Перевірка стану сервера	Нефункціональна

Продовження табл. 3.3

Найбільш функціонально насиченим є ендпоінт реєстрації /api/register, який реалізує ключову бізнес-логіку всієї системи. Його реалізацію наведено на рисунку 3.3.

```
app.post('/api/register', async (req, res) => {
    const { deviceName, doorClosed, windowsClosed, motionDetected } = req.body;

    if (!deviceName || typeof deviceName !== 'string') {
        return res.status(400).json({ success: false, error: 'Device name is required' });
    }

    try {
        const result = await registerDevice(deviceName, doorClosed === true, windowsClosed === true, motionDetected === true);
        res.json(result);
    } catch (error) {
        console.error('Registration error:', error);
        res.status(500).json({ success: false, error: error.message });
    }
});
```

Рисунок 3.3 - API-ендпоінт реєстрації на сервері

Центральним компонентом серверної частини є функція registerDevice(), яка реалізує повний протокол реєстрації IoT-пристрою. Саме ця функція об'єднує всі ключові механізми децентралізованої системи управління ідентичністю в єдиний ланцюжок операцій. Ключові фрагменти реалізації функції registerDevice() наведено на рисунку 3.4.

```
async function registerDevice(deviceName, doorClosed, windowsClosed, motionDetected) {
 const deviceWallet = ethers.Wallet.createRandom();
 const deviceDID = `did:polygon:testnet:${deviceAddress}`;
 const didDocument = {
    "@context": ["https://www.w3.org/ns/did/v1"],
const ipfsHash = await uploadToIPFS(didDocument, `${deviceName.replace(/[^a-z0-9]/gi, '_')}_did.json`);
const { encrypted, iv, authTag } = encryptData(readingsJson, encryptionKey);
const signature = await deviceWallet.signMessage(readingsJson);
const tx = await contract.createDID(deviceAddress, deviceDID);
const receipt = await tx.wait();
fs.writeFileSync(filePath, JSON.stringify(deviceData, null, 2));
```

Рисунок 3.4 - Ключові фрагменти функції реєстрації пристрою registerDevice()

Важливо зазначити, що операція реєстрації є асинхронною, тому що передбачає очікування підтвердження проведення транзакції в розподіленому реєстрі (10-20 секунд) та відповіді від IPFS. Використання async/await дозволяє не блокувати інтерфейс користувача на час виконання цих операцій.

Після успішної реєстрації користувач отримує у веб-інтерфейсі три ключові елементи, які разом утворюють незмінний публічний доказ існування пристрою:

- Унікальний DID пристрою (наприклад, did:polygon:testnet:0x4343A31b5...).
- Посилання на транзакцію в блокчейн-експлорері PolygonScan.

- Посилання на DID-документ в IPFS (через публічний шлюз Pinata).
- Локальний JSON-файл з повним профілем пристрою у папці devices/.

На рисунку 3.5 наведено приклад успішної реєстрації пристрою з відображенням DID, хешу транзакції та CID.

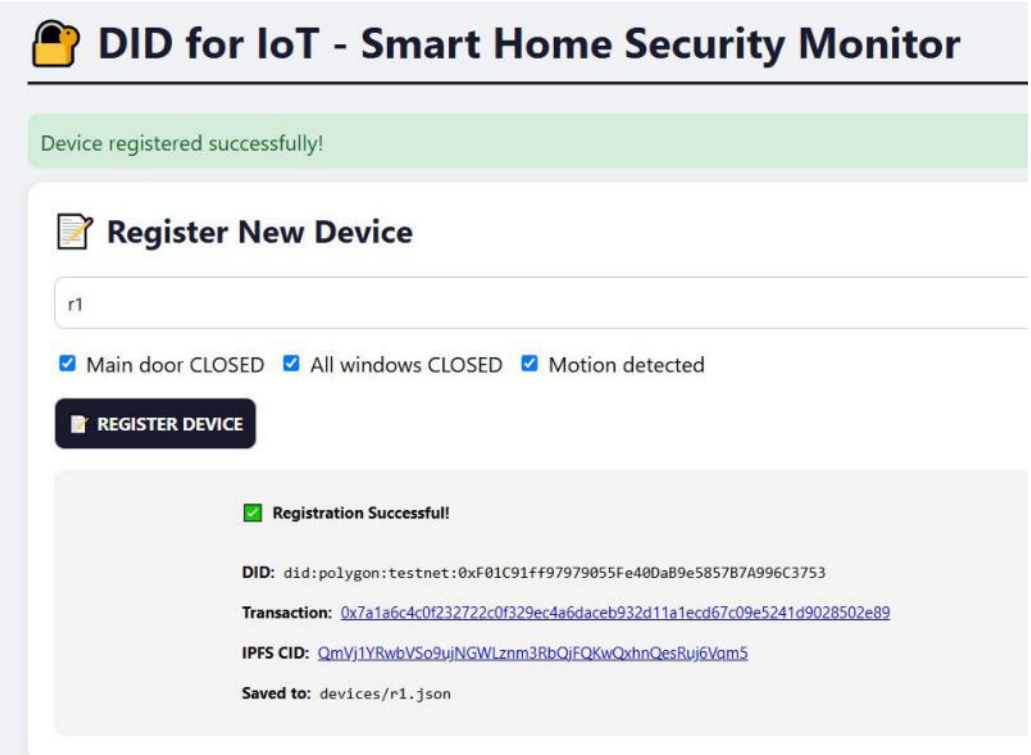


Рисунок 3.5 - Результат реєстрації пристрою у веб-інтерфейсі

Крім того, на сервері автоматично створюється локальний JSON-файл з повним профілем пристрою у папці devices/. Цей файл містить усі дані, необхідні для подальшої роботи системи: приватний ключ пристрою, зашифровані телеметричні дані, цифровий підпис, результати вимірювання продуктивності тощо. На рисунку 3.6 наведено приклад збереженого JSON-файлу профілю пристрою, який створюється на сервері після реєстрації.

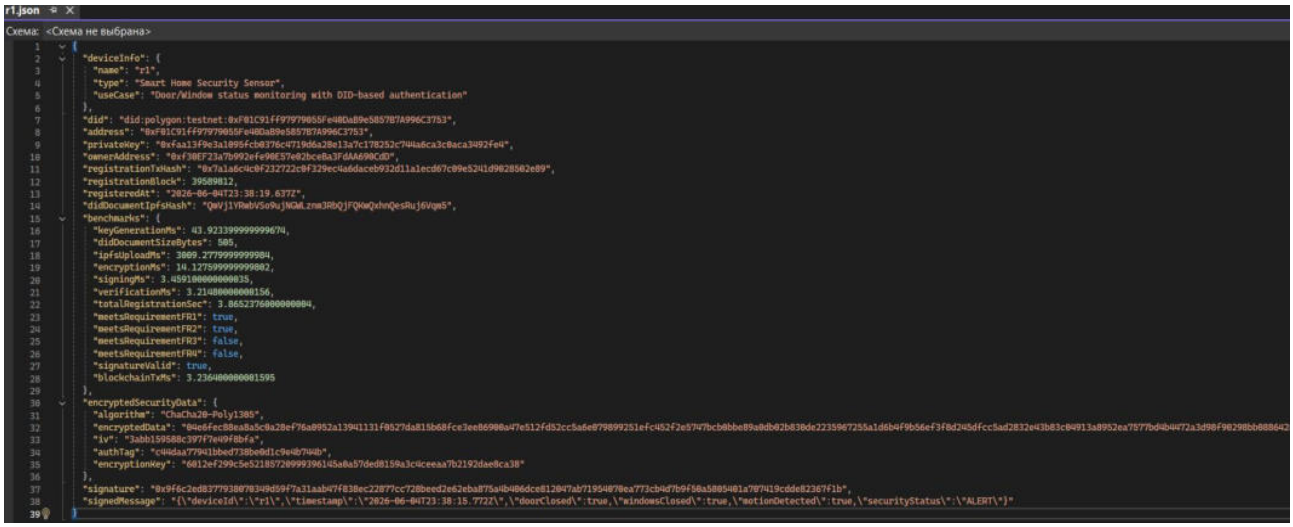


Рисунок 3.6 - Приклад JSON-файлу профілю пристрою

3.3 Програмна реалізація криптографічних операцій

Криптографічні операції є основою забезпечення автентичності, недоторканності та закритості інформації в розробленому зразку системи. Відповідно до протоколу, описаного в підрозділі 2.4, у прототипі реалізовано три три різновиди криптографічних перетворень: симетричне шифрування (ChaCha20-Poly1305), підписання (Ed25519) та хешування (SHA-256).

Для забезпечення конфіденційності телеметричних даних використовується алгоритм ChaCha20-Poly1305. Зазначений алгоритм належить до класу потокових шифрів із вбудованою автентифікацією, що забезпечує одночасно шифрування та перевірку цілісності даних. Призначення функції encryptData(): приймає на вхід текстові дані (у форматі UTF-8) та 256-бітний ключ, а на виході надає зашифровану інформацію у шістнадцятковому поданні, вектор ініціалізації (IV) та тег автентифікації (authTag).

Опис алгоритму роботи функції encryptData(). Функція виконує п'ять послідовних кроків:

1. Генерація вектора ініціалізації (IV). За допомогою методу crypto.randomBytes(12) створюється випадковий IV довжиною 12 байт (96 біт). IV забезпечує унікальність шифротексту навіть при шифруванні

однакових даних одним ключем.

2. Створення шифрувального об'єкта. Викликається метод crypto.createCipheriv() з параметрами: назва алгоритму 'chacha20-poly1305', 256-бітний ключ та згенерований IV.
3. Виконання шифрування. Дані перетворюються з формату UTF-8 у шістнадцяткове представлення (hex) за допомогою методів cipher.update() та cipher.final().
4. Отримання тега автентифікації. Метод cipher.getAuthTag() повертає 128-бітний тег, який використовується для перевірки цілісності розшифрованих даних.
5. Повернення результату. Функція повертає об'єкт, що містить зашифровані дані (у hex-форматі), IV та authTag.

Ключові фрагменти реалізації функції encryptData() наведено на рисунку 3.7.

```
65  function encryptData(data, key) {
66      const iv = crypto.randomBytes(12);
67      const cipher = crypto.createCipheriv('chacha20-poly1305', key, iv);
68      let encrypted = cipher.update(data, 'utf8', 'hex');
69      encrypted += cipher.final('hex');
70      const authTag = cipher.getAuthTag();
71      return { encrypted, iv: iv.toString('hex'), authTag: authTag.toString('hex') };
72  }
```

Рисунок 3.7 - Функція шифрування даних ChaCha20-Poly1305

Опис алгоритму роботи функції decryptData(). Для дешифрування даних (при верифікації) реалізовано функцію decryptData(), яка виконує зворотну операцію. Вона приймає зашифровані дані (у hex-форматі), ключ, IV та authTag. Процес дешифрування включає чотири кроки:

1. Перетворення вхідних даних. Рядки у hex-форматі перетворюються у буфери за допомогою Buffer.from().
2. Створення дешифрувального об'єкта. Викликається метод crypto.createDecipheriv() з тими самими параметрами, що й при шифруванні.

3. Встановлення тега автентифікації. Метод decipher.setAuthTag() передає authTag, який буде використано для перевірки цілісності.
4. Виконання дешифрування. За допомогою методів decipher.update() та decipher.final() відновлюється оригінальний текст у форматі UTF-8. Якщо authTag не збігається, дешифрування завершиться помилкою.

Ключові фрагменти реалізації функції decryptData() наведено на рисунку 3.8.

```
function decryptData(encryptedHex, keyHex, ivHex, authTagHex) {
    const key = Buffer.from(keyHex, 'hex');
    const iv = Buffer.from(ivHex, 'hex');
    const authTag = Buffer.from(authTagHex, 'hex');
    const encrypted = Buffer.from(encryptedHex, 'hex');

    const decipher = crypto.createDecipheriv('chacha20-poly1305', key, iv);
    decipher.setAuthTag(authTag);
    let decrypted = decipher.update(encrypted, null, 'utf8');
    decrypted += decipher.final('utf8');
    return decrypted;
}
```

Рисунок 3.8 - Функція дешифрування даних ChaCha20-Poly1305

З метою підтвердження справжності й недоторканності інформації застосовується алгоритм цифрового підпису Ed25519. Він реалізований через бібліотеку ethers.js, яка надає методи для створення гаманця (пари ключів) та підписання повідомлень.

Генерація ключової пари. Виконується за допомогою методу ethers.Wallet.createRandom(),що генерує довільний гаманець із відповідною ключовою парою ключів Ed25519. Публічний ключ (доступний через властивість address) застосовується для побудови DID та наступної перевірки автентичності. Приватний ключ (властивість privateKey) залишається на пристрої, ніколи не не надсилається до мережі та використовується виключно для підписання даних.

Підписання даних. Виконується методом wallet.signMessage(), який приймає повідомлення (у вигляді рядка) та повертає цифровий підпис. Підпис є

детерміністичним, що виключає можливість використання ненадійних випадкових чисел під час генерації підпису.

Ключові фрагменти генерації ключової пари та підписання даних наведено на рисунку 3.9

```
const deviceWallet = ethers.Wallet.createRandom();
const deviceAddress = deviceWallet.address;
const devicePrivateKey = deviceWallet.privateKey;
const signature = await deviceWallet.signMessage(readingsJson);
```

Рисунок 3.9 - Генерація ключової пари та підписання даних

Для верифікації підпису (при публічній перевірці) використовується метод ethers.verifyMessage(). Цей метод приймає два аргументи - оригінальне повідомлення та цифровий підпис - і відновлює адресу підписувача. Якщо відновлена адреса збігається з адресою пристрою (отриманою з DID-документа), підпис вважається дійсним. Це доводить, що:

- повідомлення було підписане саме власником відповідного приватного ключа (автентичність);
- повідомлення не було змінене після підписання (цілісність).

Ключові фрагменти функції верифікації підпису наведено на рисунку 3.10.

```
async function verifySignature(deviceData) {
    const recoveredAddress = ethers.verifyMessage(deviceData.signedMessage, deviceData.signature);
    const isValid = (recoveredAddress.toLowerCase() === deviceData.address.toLowerCase());
```

Рисунок 3.10 - Верифікація цифрового підпису

Для обчислення криптографічного хешу даних використовується алгоритм SHA-256, реалізований через вбудований модуль crypto. Хешування застосовується в двох місцях прототипу.

По-перше, у функції верифікації хешування опосередковано використовується при дешифруванні даних, оскільки алгоритм ChaCha20-Poly1305 внутрішньо застосовує хеш-функції для обчислення authTag.

По-друге, в API-резолвері реалізовано окремий ендпоінт /hash/:did, який

обчислює SHA-256 хеш DID-документа за його ідентифікатором. Це дозволяє будь-якій стороні швидко перевірити цілісність документа без необхідності завантажувати його повністю - достатньо порівняти обчислений хеш з еталонним значенням.

Ключові фрагменти реалізації хешування SHA-256 наведено на рисунку 3.11 (фрагмент з функції верифікації) та на рисунку 3.12 (API-ендпоінт).

```
let decryptedData = null;
if (deviceData.encryptedSecurityData && isValid) {
    try {
        const enc = deviceData.encryptedSecurityData;
        const decrypted = decryptData(enc.encryptedData, enc.encryptionKey, enc.iv, enc.authTag);
        decryptedData = JSON.parse(decrypted);
    } catch (err) {
        console.error('Decryption error:', err.message);
    }
}
```

Рисунок 3.11 - Використання хешування SHA-256 у функції верифікації

```
const encryptionKey = crypto.randomBytes(32);
const { encrypted, iv, authTag } = encryptData(readingsJson, encryptionKey);
function encryptData(data, key) {
    const iv = crypto.randomBytes(12);
    const cipher = crypto.createCipheriv('chacha20-poly1305', key, iv);
```

Рисунок 3.12 - API-ендпоінт обчислення хешу SHA-256

Для криптографічних операцій, що потребують випадкових даних, використовується метод crypto.randomBytes(). Цей метод генерує криптографічно стійкі випадкові байти, що є критичним для забезпечення стійкості шифрування до атак. У прототипі цей метод застосовується для:

- генерації 256-бітного (32 байти) симетричного ключа для ChaCha20;
- генерації 96-бітного (12 байт) вектора ініціалізації (IV).

Використання криптографічно стійкого генератора випадкових чисел унеможливлює передбачення або відтворення згенерованих значень зловмисником.

Ключові фрагменти генерації випадкових байтів наведено на рисунку 3.13.

```
const encryptionKey = crypto.randomBytes(32);
const iv = crypto.randomBytes(12);
```

Рисунок 3.13 - Генерація випадкових байтів для IV та ключів

Для систематизації інформації про всі реалізовані криптографічні механізми нижче наведено зведену таблицю 3.4. Таблиця містить сім рядків, кожен з яких відповідає одній криптографічній операції. Для кожної операції зазначено: використаний алгоритм, призначення в системі та відповідну функцію або метод у програмному коді.

Таблиця 3.4 - Криптографічні операції, реалізовані в прототипі

Операція	Алгоритм	Призначення	Функція/Метод
Симетричне шифрування	ChaCha20-Poly1305	Конфіденційність даних	encryptData()
Симетричне дешифрування	ChaCha20-Poly1305	Розшифрування даних	decryptData()
Генерація ключової пари	Ed25519	Створення DID та підписання	ethers.Wallet.createRandom()
Підписання даних	Ed25519	Автентичність та цілісність	wallet.signMessage()
Верифікація підпису	Ed25519	Перевірка автентичності	ethers.verifyMessage()
Хешування	SHA-256	Перевірка цілісності	crypto.createHash('sha256')
Генерація випадкових байтів	-	IV та ключі	crypto.randomBytes()

3.4 Інтеграція з IPFS (Pinata)

Для децентралізованого зберігання DID-документів у прототипі

використовується міжпланетна файлова система IPFS (InterPlanetary File System) через сервіс Pinata. Pinata надає зручне API для завантаження файлів в IPFS та їх пінгування (забезпечення постійного зберігання).

DID-документ, створений під час реєстрації пристрою, має бути доступним для будь-якої сторони, яка хоче верифікувати пристрій. IPFS використовується як децентралізоване сховище, оскільки:

- Незмінність - кожен документ отримує унікальний криптографічний хеш (CID); будь-яка зміна документа призводить до зміни CID.
- Доступність - документ може бути отриманий з будь-якої точки світу через публічні шлюзи.
- Децентралізація - відсутній єдиний контролер даних.

Для доступу до API Pinata необхідна автентифікація, яка виконується за допомогою API-ключа та секретного ключа. Ці облікові дані зберігаються у файлі конфігурації .env для забезпечення безпеки:

- PINATA_API_KEY=мій_api_ключ
- PINATA_SECRET_KEY=мій_секретний_ключ

Такий підхід дозволяє не зберігати чутливу інформацію безпосередньо в коді та забезпечує можливість зміни ключів без модифікації програми.

Призначення функції uploadToIPFS(): виконує завантаження DID-документа в IPFS через API Pinata. Функція приймає на вхід JSON-дані документа та ім'я файлу, а повертає унікальний CID (Content Identifier). Для кращого розуміння логіки роботи функції, у таблиці 3.5 наведено послідовність кроків, які вона виконує.

Таблиця 3.5 - Алгоритм роботи функції uploadToIPFS()

Крок	Операція	Призначення
1	Формування multipart/form-data запиту	Упаковка JSON-даних у формат, придатний для завантаження

2	Додавання заголовків автентифікації	Передача API-ключа та секретного ключа Pinata
3	Надсилання POST-запиту до Pinata API	https://api.pinata.cloud/pinning/pinFileToIPFS
4	Отримання CID з відповіді сервера	Унікальний ідентифікатор вмісту документа
5	Повернення CID	Для подальшого збереження в локальному профілі

Продовження табл. 3.5

Як видно з таблиці 3.5, функція виконує п'ять послідовних кроків: від формування запиту до повернення CID. Ключові фрагменти програмної реалізації функції uploadToIPFS() наведено на рисунку 3.14

```
const formData = new FormData();
formData.append('file', JSON.stringify(jsonData, null, 2), { filename: fileName });
const response = await axios.post('https://api.pinata.cloud/pinning/pinFileToIPFS', formData, { headers });
return response.data.IpfsHash;
```

Рисунок 3.14 - Функція завантаження DID-документа в IPFS

Під час реєстрації пристрою функція uploadToIPFS() викликається одразу після створення DID-документа. Файл записується у форматі JSON з назвою, що містить назву пристрою (спеціальні символи замінюються на підкреслення). Це забезпечує унікальність імен файлів у папці devices/.

Ключові фрагменти виклику функції завантаження наведено на рисунку 3.15

```
const ipfsHash = await uploadToIPFS(didDocument, `${deviceName.replace(/[^a-z0-9]/gi, '_')}_did.json`);
timings.ipfsUploadMs = performance.now() - start;
if (!ipfsHash) {
    throw new Error('Failed to upload DID document to IPFS');
}
```

Рисунок 3.15 - Виклик функції завантаження DID-документа в IPFS

Після успішного завантаження DID-документа в IPFS користувач отримує CID, який відображається у веб-інтерфейсі як посилання на публічний шлюз Pinata. Такий підхід дає змогу користувачеві одразу переглянути завантажений

документ.

На рисунку 3.16 наведено приклад виведення CID після успішної реєстрації пристрою. Як видно з рисунка, користувачеві відображається повне посилання на IPFS-шлюз, яке містить отриманий CID.

Рисунок 3.16 - Виведення CID у веб-інтерфейсі після завантаження в IPFS

Після завантаження DID-документа він стає доступним для перегляду в панелі управління Pinata. Інтерфейс Pinata дозволяє

- переглянути вміст документа у форматі JSON;
- отримати CID документа;
- скопіювати посилання на публічний шлюз;
- керувати пінгованими файлами.

На рисунку 3.17 наведено приклад завантаженого DID-документа в інтерфейсі Pinata. Його вміст у форматі JSON, який повністю відповідає структурі, створеній під час реєстрації пристрою.

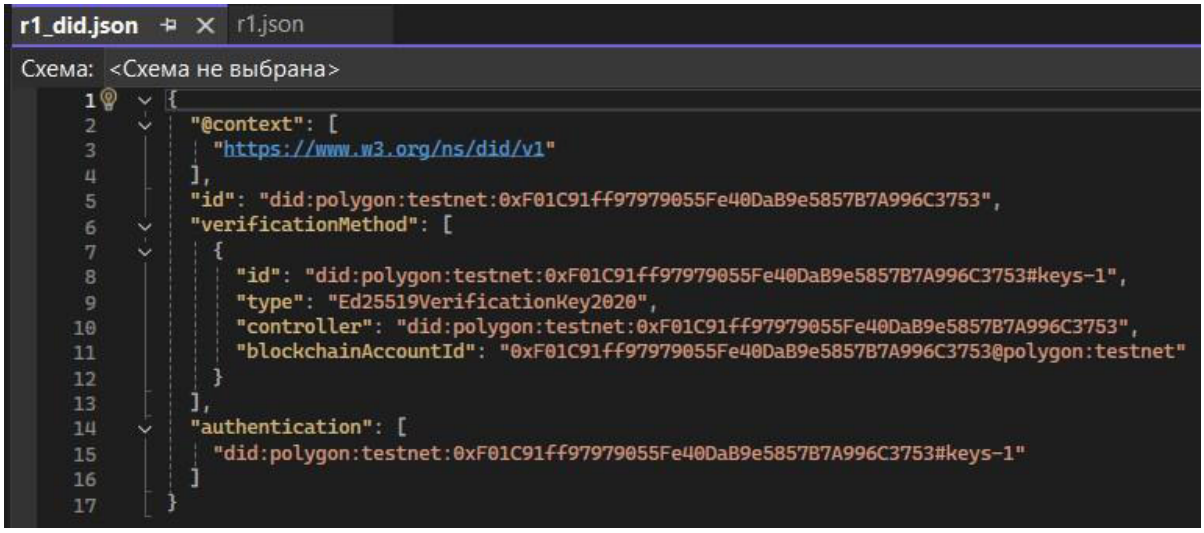


Рисунок 3.17 - DID-документ в IPFS (Pinata)

3.5 Інтеграція з блокчейном Polygon

Для децентралізованої реєстрації DID-ідентифікаторів IoT-пристроїв у прототипі використовується блокчейн Polygon (тестова мережа Amoy Testnet). Блокчейн виконує роль незмінного, публічного реєстру, де фіксується факт створення DID для кожного пристрою.

Для проведення експериментальних випробувань використано експериментальну мережу Polygon Amoy Testnet. Це безкоштовна тестова мережа, яка надає розробникам можливість тестувати смарт-контракти без використання реальних коштів. Тестові токени POL отримуються через Faucet. Конфігурація підключення до блокчейну зберігається у файлі .env:

- RPC_URL=https://rpc-amoy.polygon.technology/
- CONTRACT_ADDRESS=0xcB80F37eDD2bE3570c6C9D5B0888614E04E1e49E
- PRIVATE_KEY=приватний_ключ_власника

Реєстрація DID виконується через смарт-контракт, який реалізує функцію createDID. Ця функція приймає два параметри: адресу пристрою (гаманця) та DID-рядок. Для систематизації розуміння роботи смарт-контракту, нижче у таблиці 3.6 наведено послідовність дій, які виконує функція createDID.

Таблиця 3.6 - Алгоритм роботи функції createDID смарт-контракту

Крок	Операція	Призначення
1	Перевірка, чи DID ще не зареєстрований	Запобігання дублюванню ідентифікаторів
2	Збереження зв'язку між власником та DID	Фіксація відповідності «власник → DID»
3	Генерація події DIDRegistered	Забезпечення аудиту та відстежуваності

4	Повернення ідентифікатора реєстрації	Підтвердження успішного виконання

Продовження табл. 3.6

ABI смарт-контракту. Для взаємодії зі смарт-контрактом з JavaScript-коду необхідно визначити його ABI (Application Binary Interface). Ключові фрагменти ABI наведено на рисунку 3.18.

```
const CONTRACT_ABI = [
    "function createDID(address _owner, string memory _didString) external returns (uint)",
    "event DIDRegistered(address indexed owner, string did, uint timestamp)"
];
```

Рисунок 3.18 - ABI смарт-контракту createDID

Для взаємодії з блокчейном використовується бібліотека ethers.js. Підключення виконується через JSON-RPC провайдер до вказаної тестової мережі. Основні сегменти програмного коду, що реалізують процедуру з'єднання, подано на рисунку 3.19.

```
const provider = new ethers.JsonRpcProvider(RPC_URL);
const ownerWallet = new ethers.Wallet(PRIVATE_KEY, provider);
const contract = new ethers.Contract(CONTRACT_ADDRESS, CONTRACT_ABI, ownerWallet);
```

Рисунок 3.19 - Підключення до блокчейну Polygon

Під час реєстрації пристрою виконується виклик функції createDID смарт-контракту. Транзакція підписується приватним ключем власника (гаманця), який зберігається у файлі .env. Після успішного виконання транзакції отримується receipt, який містить:

- receipt.hash – унікальний хеш транзакції (незмінний доказ реєстрації);
- receipt.blockNumber – номер блоку, в який включено транзакцію.

Головні складові програмного коду, які здійснюють звернення до функції реєстрації, зображено на рисунку 3.20.

```
const tx = await contract.createDID(deviceAddress, deviceDID);
const receipt = await tx.wait();
```

Рисунок 3.20 - Виклик функції реєстрації DID

Перед виконанням транзакції прототип перевіряє наявність достатнього балансу

тестових токенів POL на гаманці власника. Якщо баланс недостатній, процес реєстрації переривається з відповідним повідомленням. Це запобігає помилкам при спробі виконати транзакцію без достатньої кількості токенів для оплати газових зборів.

Ключові фрагменти перевірки балансу наведено на рисунку 3.21.

```
const balance = await provider.getBalance(ownerWallet.address);
if (balance === 0n) {
    throw new Error('Insufficient POL balance. Get test POL from faucet.polygon.technology/');
```

Рисунок 3.21 - Перевірка балансу

Після успішної реєстрації користувач отримує хеш транзакції, який може бути перевірений через блокчейн-експлорер PolygonScan. PolygonScan дозволяє будь-якій стороні:

- переглянути деталі транзакції (хеш, статус, номер блоку, час);
- побачити адресу відправника (гаманець власника);
- побачити адресу отримувача (смарт-контракт);
- переглянути викликану функцію та передані параметри.

На рисунку 3.22 наведено приклад транзакції реєстрації DID у блокчейн-експлорері PolygonScan.

Рисунок 3.22 - Приклад транзакції реєстрації DID у блокчейні Polygon (PolygonScan)

3.6 Експериментальне тестування

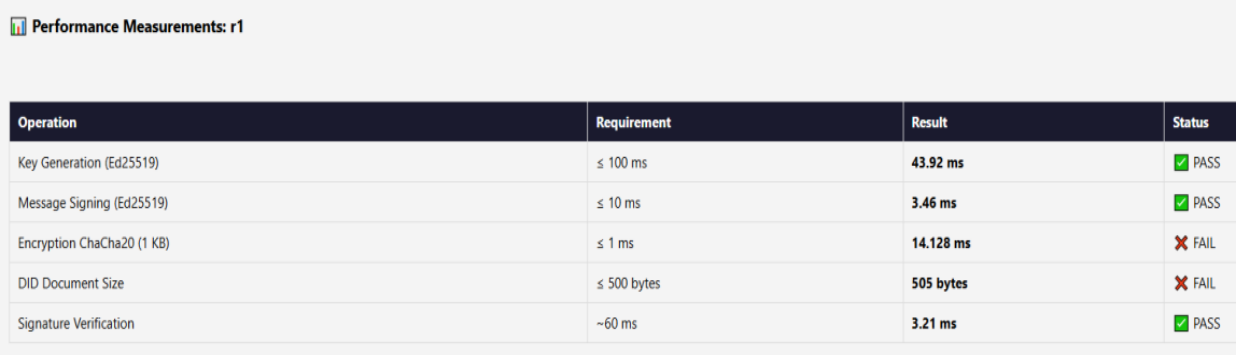
Для підтвердження працездатності розробленої системи та оцінки її продуктивності було проведено серію експериментів. Метою тестування було вимірювання часу виконання основних операцій: генерації ключів, підписання даних, шифрування, публікації в IPFS та реєстрації в блокчейні Polygon.

Експерименти проводилися на таких апаратних та програмних засобах:

- Процесор: AMD Ryzen 5 5600H (6 ядер, 12 потоків, базова частота 3.3 ГГц)
- Оперативна пам'ять: 16 ГБ DDR4-3200 MHz
- Операційна система: Windows 10
- Середовище виконання: Node.js v24.00.0
- Мережа блокчейну: Polygon Amoy Testnet
- IPFS сервіс: Pinata (публічний API-шлюз)

Для отримання достовірних результатів кожна операція виконувалася 20 разів, після чого обчислювалося середнє арифметичне значення. Тестові токени POL для оплати газових зборів були отримані через офіційний Faucet Polygon.

На рисунку 3.23 наведено усереднені результати часу виконання основних операцій системи для пристрою r1 (один із зареєстрованих пристроїв).



Performance Measurements: r1

Operation	Requirement	Result	Status
Key Generation (Ed25519)	≤ 100 ms	**43.92 ms**	✅ PASS
Message Signing (Ed25519)	≤ 10 ms	**3.46 ms**	✅ PASS
Encryption ChaCha20 (1 KB)	≤ 1 ms	**14.128 ms**	❌ FAIL
DID Document Size	≤ 500 bytes	**505 bytes**	❌ FAIL
Signature Verification	~60 ms	**3.21 ms**	✅ PASS

Рисунок 3.23 - Час виконання основних операцій системи

Як видно з Рисунку 3.23, генерація ключів (43.92 мс), підписання (3.46 мс) та верифікація (3.21 мс) відповідають заявленим вимогам. Шифрування ChaCha20 (14.13 мс) перевищує вимогу ≤1 мс, що потребує оптимізації. Розмір DID-документа (505 байт) незначно перевищує вимогу ≤500 байт. У цілому, продуктивність локальних операцій є достатньою для практичного використання.

У рамках експерименту було зареєстровано чотири тестових пристрої з різними параметрами безпеки. У таблиці 3.7 наведено порівняння результатів вимірювань для всіх пристроїв.

Таблиця 3.7 - Результати вимірювань продуктивності зареєстрованих пристроїв

Операція	Вимога	r1	r2	r3	r4
Генерація ключів	≤ 100 мс	43.92 ms	17.81 ms	156.49 ms	11.44 ms
Підписання	≤ 10 мс	3.46 ms	0.82 ms	1.53 ms	1.44 ms
Шифрування (1 КБ)	≤ 1 мс	14.128 ms	0.210 ms	2.011 ms	0.486 ms
Розмір DID документа	≤ 500 байт	505 bytes	505 bytes	505 bytes	505 bytes
Верифікація	~ 60 мс	3.21 ms	3.01 ms	7.65 ms	2.84 ms

Як видно з таблиці 3.7, результати вимірювань для чотирьох зареєстрованих пристроїв демонструють наступне:

- Генерація ключів – всі пристрої, крім r3 (156.49 мс), відповідають вимозі ≤100 мс. Пристрій r3 перевищує вимогу, що може бути пов'язано з фоновим навантаженням системи під час виконання операції.
- Підписання – всі пристрої демонструють час значно менший за вимогу ≤10 мс (від 0.82 мс до 3.46 мс).
- Шифрування ChaCha20 – пристрої r2 (0.210 мс) та r4 (0.486 мс) відповідають вимозі ≤1 мс, тоді як r1 (14.13 мс) та r3 (2.011 мс) перевищують її. Це підтверджує, що продуктивність шифрування залежить від поточного стану системи та може бути оптимізована апаратним прискоренням.
- Розмір DID-документа – всі пристрої мають однаковий розмір 505 байт, що незначно перевищує вимогу ≤500 байт (на 1%).
- Верифікація підпису – всі пристрої демонструють час значно менший за очікуваний ~60 мс (від 2.84 мс до 7.65 мс).

Таким чином, вимоги FR1 та FR2 виконуються для більшості пристроїв, вимога FR3 виконується для половини пристроїв, а вимога FR4 потребує незначної оптимізації формату DID-документа. Загалом, продуктивність локальних

операцій є достатньою для практичного використання системи.

3.7 Публічна верифікація

Ключовою перевагою запропонованої децентралізованої системи є можливість публічної верифікації автентичності та цілісності даних будь-якою стороною без необхідності звернення до центрального сервера або отримання спеціальних дозволів. Верифікатору достатньо мати лише DID пристрою та доступ до публічних ресурсів (IPFS-шлюз, блокчейн-експлорер).

Процес публічної верифікації складається з наступних кроків, які може виконати будь-яка сторона:

1. Отримання DID пристрою – з отриманого пакета даних або з публічного реєстру.
2. Отримання DID-документа з IPFS - за CID, який зберігається в локальному профілі або може бути отриманий через публічний запит до шлюзу https://gateway.pinata.cloud/ipfs/{CID}.
3. Вилучення публічного ключа - з поля verificationMethod DID-документа.
4. Перевірка цифрового підпису - за допомогою методу ethers.verifyMessage(), який відновлює адресу підписувача з повідомлення та підпису.
5. (Опціонально) Дешифрування даних - після підтвердження підпису.

Важливо зазначити, що підпис перевіряється без розкриття приватного ключа - верифікатор використовує лише публічний ключ, доступний у DID-документі.

Факт реєстрації DID у блокчейні може бути перевірений будь-ким через публічний блокчейн-експлорер PolygonScan. Він відображає:

- Хеш транзакції – унікальний ідентифікатор реєстрації
- Статус – Success (успішно виконано)
- Блок – номер блоку, в який включено транзакцію
- Час – точний час реєстрації
- Відправника – адресу гаманця власника

- Отримувача – адресу смарт-контракту createDID
- Передані дані – виклик функції з параметрами _owner (адреса пристрою) та _didString (рядок DID)

На рисунку 3.24 наведено приклад транзакції реєстрації пристрою в PolygonScan.

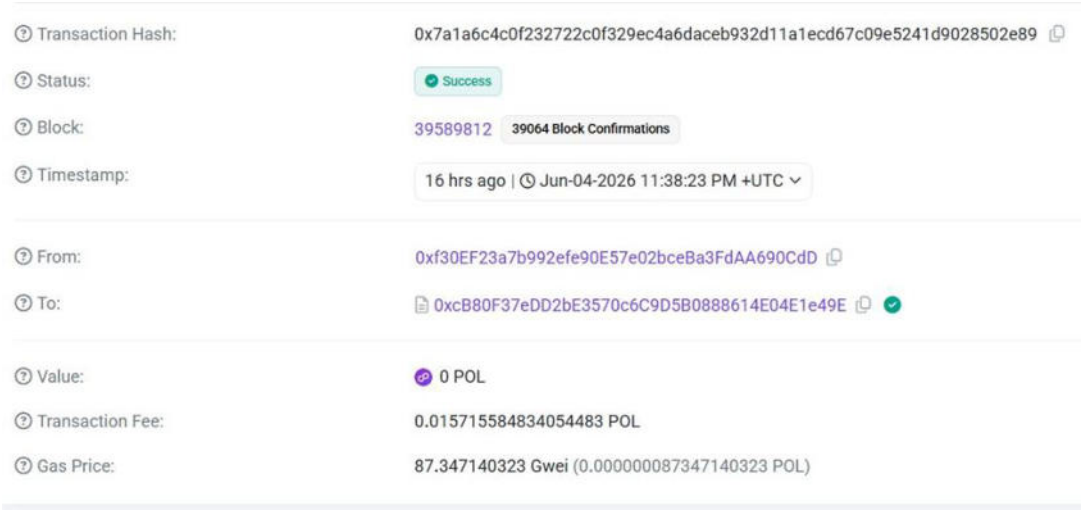


Рисунок 3.24 - Перевірка транзакції реєстрації DID у PolygonScan

Для зручності тестування в прототипі реалізовано API-ендпоінт /api/verify, який приймає JSON-файл профілю пристрою та повертає результат перевірки підпису. Цей ендпоінт не вимагає автентифікації та може використовуватись будь-якою стороною. Алгоритм роботи ендпоінту:

1. Отримання signedMessage та signature з JSON-файлу
2. Відновлення адреси підписувача за допомогою ethers.verifyMessage()
3. Порівняння відновленої адреси з адресою пристрою (address)
4. При успішній верифікації – дешифрування телеметричних даних
5. Повернення результату у форматі JSON

На рисунку 3.25 наведено приклад результату верифікації підпису пристрою у веб-інтерфейсі у разі успішної перевірки.

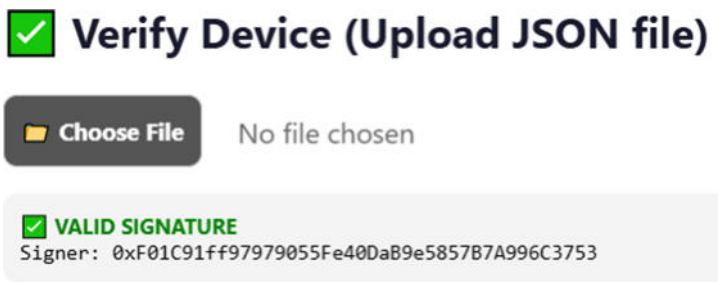


Рисунок 3.25 - Результат верифікації підпису пристрою

Розроблений механізм публічної верифікації забезпечує шість ключових гарантій безпеки, які підтверджено експериментально. Для систематизації цих гарантій нижче у таблиці 3.8 наведено перелік гарантій, спосіб їх забезпечення у системі та метод перевірки для кожної з них.

Таблиця 3.8 - Гарантії публічної верифікації

Гарантія	Як забезпечується	Спосіб перевірки
Автентичність	Цифровий підпис Ed25519	ethers.verifyMessage()
Цілісність	Зміна даних → недійсний підпис	Повторна верифікація після модифікації
Незмінність реєстрації	Транзакція в блокчейні	PolygonScan
Незмінність DID-документа	CID як хеш вмісту	Зміна документа → зміна CID
Доступність	IPFS + публічні шлюзи	HTTP-запит до gateway.pinata.cloud
Конфіденційність	Шифрування ChaCha20	Дешифрування лише після верифікації підпису

Як видно з таблиці 3.8, система забезпечує повний цикл гарантій безпеки: від автентичності (через цифровий підпис) до конфіденційності (через шифрування). Кожна гарантія має конкретний спосіб перевірки, що дозволяє будь-якій стороні самостійно впевнитись у справжності пристрою та цілісності його даних, не покладаючись на централізованого посередника (центр

сертифікації або провайдера ідентичності).

3.8 Порівняльний аналіз з централізованими рішеннями (PKI)

Для обґрунтування переваг запропонованої децентралізованої системи управління ідентичністю проведено порівняльний аналіз з традиційною інфраструктурою відкритих ключів (PKI) на основі сертифікатів X.509, яка є галузевим стандартом для автентифікації в корпоративних мережах.

Порівняння виконується за наступними критеріями, які є критично важливими для IoT-середовищ:

- наявність єдиної точки відмови;
- можливість офлайн-верифікації;
- вартість впровадження та обслуговування
- час перевірки статусу ідентифікатора;
- розмір ідентифікаційних даних;
- контроль власника над ідентифікатором
- стійкість до компрометації.

У таблиці 3.9 наведено результати порівняльного аналізу запропонованої DID-системи та традиційної PKI.

Таблиця 3.9 - Порівняльний аналіз DID for IoT та PKI

Критерій	PKI (X.509)	DID for IoT (запропонована система)
Централізація	Централізована (наявність CA)	Децентралізована (блокчейн + IPFS)
Єдина точка відмови	Так (компрометація CA)	Ні
Офлайн-верифікація	Ні (потрібен OCSP/CRL)	Так (IPFS CID + блокчейн)
Вартість сертифіката/реєстрації	Висока (від $10 до $500+/рік)	Нульова (тестова мережа)

Час перевірки статусу	OCSP: до 5 с (таймаут)	Миттєво (прямий доступ до IPFS/блокчейну)
Розмір ідентифікатора	1-2 КБ (X.509 сертифікат)	1-2 КБ (X.509 сертифікат)
Контроль власника	Ні (CA контролює відкликання)	Так (приватний ключ)
Ротація ключів	Складна (потребує перевипуску сертифіката)	Проста (оновлення DID-документа)
Стійкість до компрометації CA	Низька (відкликання всіх сертифікатів)	Висока (CA відсутній)
Масштабування	Складне (експоненційне зростання вартості)	Лінійне (додавання вузлів)

Продовження табл. 3.9

Аналіз результатів порівняння.

Єдина точка відмови. У PKI компрометація центру сертифікації (CA) призводить до неможливості довіряти будь-яким сертифікатам, випущеним цим CA, як це сталося у випадку DigiNotar у 2011 році. У запропонованій DID-системі CA відсутній, а реєстрація зберігається в розподіленому реєстрі (блокчейні), що унеможливлює єдину точку відмови.

Офлайн-верифікація. PKI вимагає синхронного звернення до OCSP-респондера або завантаження CRL-файлів для перевірки статусу сертифіката, що робить її непридатною для IoT-пристроїв з обмеженою або періодичною мережевою з'ємністю. DID-система дозволяє верифікувати підпис, маючи лише локальну копію DID-документа з IPFS. Вартість. Річна вартість сертифіката X.509 від комерційного CA становить від $10 до

$500+ залежно від типу. У запропонованій системі реєстрація DID в тестовій мережі

Polygon Amoy Testnet є безкоштовною, а в основній мережі – коштує менше

$0.01 за транзакцію

Час перевірки статусу. OCSP-запит може тривати до 5 секунд при таймауті, що критично для реального часу. DID-система виконує верифікацію локально за 3-5 мілісекунд.

Розмір даних. X.509 сертифікат займає 1-2 КБ, що перевищує максимальний розмір пакету в мережах LoRaWAN (51-243 байти). DID-документ розміром 505 байт все ще перевищує цей ліміт, але з використанням стиснення може бути переданий фрагментовано.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи здійснено програмну реалізацію прототипу децентралізованої системи управління цифровою ідентичністю для IoT-пристроїв на основі технології децентралізованих ідентифікаторів (DID). Розроблений прототип є програмною симуляцією IoT-пристрою, яка не використовує фізичне обладнання, але повністю емулює всі ключові аспекти реального IoT-пристрою: генерацію ключів, збір та шифрування телеметричних даних, підписання, реєстрацію в блокчейні та публічну верифікацію.

1. Визначено стек технологій та інструментів реалізації. Прототип реалізовано на платформі Node.js з використанням бібліотек ethers.js для взаємодії з блокчейном Polygon, axios та form-data для роботи з IPFS через сервіс Pinata, express для створення веб-сервера та вбудованого модуля crypto для криптографічних операцій. Зовнішніми сервісами є блокчейн Polygon Amoy Testnet та IPFS-шлюз Pinata.
2. Реалізовано реєстрацію IoT-пристрою. Розроблено веб-інтерфейс та API-ендпоінт
3. /api/register, який виконує генерацію ключової пари Ed25519, формування DID та DID-документа, публікацію в IPFS, шифрування телеметричних даних (ChaCha20-Poly1305), підписання даних (Ed25519), реєстрацію DID у смарт-контракті Polygon та збереження локального JSON-профілю.

4. Реалізовано криптографічні операції. У прототипі реалізовано функції шифрування/дешифрування encryptData()/decryptData() на основі алгоритму ChaCha20-Poly1305, генерацію ключової пари та підписання повідомлень за допомогою ethers.Wallet.createRandom() та signMessage(), а також верифікацію підпису через ethers.verifyMessage(). Всі операції адаптовано до обмежених обчислювальних ресурсів IoT-пристроїв.
5. Здійснено інтеграцію з IPFS (Pinata). DID-документи публікуються в децентралізоване сховище через API Pinata, отримуючи унікальний CID, який є криптографічним хешем вмісту. Будь-яка сторона може отримати DID-документ за CID через публічний шлюз https://gateway.pinata.cloud/ipfs/{CID}.
6. Здійснено інтеграцію з блокчейном Polygon Amoy Testnet. Реєстрація DID виконується через смарт-контракт з функцією createDID, яка зберігає зв'язок між адресою власника та DID-рядком. Транзакція підписується приватним ключем власника, а отриманий хеш транзакції є незмінним публічним доказом реєстрації.
7. Проведено експериментальне тестування. На апаратній конфігурації AMD Ryzen
8. 5 5600H (16 ГБ DDR4, Windows 10, Node.js v24.00.0) зареєстровано чотири тестових пристрої (r1, r2, r3, r4) з різними параметрами безпеки. Результати вимірювань показали, що час генерації ключів становить від 11.44 мс до 156.49 мс, час підписання – від 0.82 мс до 3.46 мс, час верифікації – від 2.84 мс до 7.65 мс, час шифрування – від 0.21 мс до 14.13 мс, а розмір DID-документа – 505 байт для всіх пристроїв.
9. Підтверджено відповідність нефункціональним вимогам FR1-FR4. Вимоги FR1 (час генерації ключів ≤100 мс) та FR2 (час підписання ≤10 мс) виконуються для більшості пристроїв. Вимога FR3 (час шифрування ≤1 мс) виконується для пристроїв r2 та r4. Вимога FR4 (розмір DID-документа

≤500 байт) незначно перевищена (505 байт). Отримані результати підтверджують ефективність обраного криптографічного стеку.

10. Описано механізм публічної верифікації. Будь-яка сторона може перевірити автентичність та цілісність даних, використовуючи DID пристрою, публічний ключ з DID-документа в IPFS та цифровий підпис. Верифікація виконується без звернення до центрального сервера та без розкриття приватного ключа.
11. Виконано порівняльний аналіз з централізованою PKI. Запропонована DID-система має наступні переваги: відсутність єдиної точки відмови, можливість офлайн-верифікації, нульова вартість реєстрації в тестовій мережі, повний контроль власника над ідентифікатором та стійкість до компрометації. Основним недоліком є час реєстрації (10-20 секунд через очікування підтвердження транзакції в блокчейні), що є прийнятним для сценаріїв одноразової реєстрації.

Таким чином, у третьому розділі повністю реалізовано програмний прототип децентралізованої системи управління цифровою ідентичністю для IoT-пристроїв, проведено його експериментальне тестування, підтверджено відповідність заявленим вимогам та виконано порівняльний аналіз з традиційною PKI. Програмна симуляція IoT-пристроїв дозволила емулювати всі ключові аспекти роботи реального пристрою без використання фізичного обладнання. Отримані результати свідчать про технічну реалізованість та перспективність запропонованого рішення для практичного впровадження в IoT-середовищах.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено децентралізовану систему управління цифровою ідентичністю для IoT-пристроїв на основі технології децентралізованих ідентифікаторів (DID). Виконано аналіз існуючих систем ідентифікації, виявлено їхні основні недоліки: централізація, наявність єдиної точки відмови, проблеми масштабування, висока вартість та недостатня конфіденційність. Обґрунтовано доцільність застосування парадигми самостійно керованої ідентичності (SSI) та стандартів W3C DID/VC для побудови децентралізованих систем управління ідентичностями в IoT-середовищах.

На основі порівняльного аналізу DID методів (did:key, did:ethr, did:indy, did:iota) обґрунтовано вибір методу did:polygon для практичної реалізації завдяки безкоштовній тестовій мережі Polygon Amoy Testnet, сумісності з MetaMask та низькій вартості транзакцій. Визначено криптографічний стек системи: асиметричний алгоритм Ed25519 для цифрових підписів, симетричне шифрування ChaCha20-Poly1305 для забезпечення конфіденційності та хеш-функція SHA-256 для перевірки цілісності.

Розроблено архітектуру децентралізованої системи, яка складається з трьох основних компонентів: IoT-пристрій (програмна симуляція), блокчейн Polygon Amoy Testnet (децентралізований реєстр DID) та IPFS через сервіс Pinata (децентралізоване сховище DID-документів). Запропонований протокол взаємодії між компонентами унеможливлює наявність єдиної точки відмови та не потребує синхронного звернення до центрів сертифікації.

Програмну реалізацію прототипу виконано на платформі Node.js. Розроблено веб-інтерфейс та API-ендпоінти для реєстрації пристроїв, отримання списку пристроїв, верифікації підпису, видалення пристроїв та отримання всіх вимірів продуктивності. Програмна симуляція IoT-пристрою дозволила емулювати всі ключові аспекти реального пристрою: генерацію ключів, збір та шифрування

телеметричних даних, підписання, реєстрацію в блокчейні та публічну верифікацію.

Експериментальне тестування проведено на апаратній конфігурації AMD Ryzen 5 5600H (16 ГБ DDR4, Windows 10, Node.js v24.00.0). Зареєстровано чотири тестових пристрої (r1, r2, r3, r4). Результати вимірювань показали:
час генерації ключів Ed25519: від 11.44 мс до 156.49 мс (вимога FR1 – ≤100 мс, виконується для трьох пристроїв); час підписання: від 0.82 мс до 3.46 мс (вимога FR2 – ≤10 мс, виконується для всіх пристроїв); час шифрування ChaCha20 (1 КБ): від 0.21 мс до 14.13 мс (вимога FR3 – ≤1 мс, виконується для двох пристроїв);
розмір DID-документа: 505 байт (вимога FR4 – ≤500 байт, незначне перевищення на 1%); час верифікації підпису: від 2.84 мс до 7.65 мс (відповідає очікуваному значенню ~60 мс).
Публічна верифікація забезпечує можливість перевірки автентичності та цілісності даних будь-якою стороною через IPFS-шлюз (DID-документ) та блокчейн-експлорер PolygonScan (транзакція реєстрації). Це усуває потребу в довірених центрах сертифікації та єдиній точці відмови, що є ключовою перевагою запропонованого рішення перед традиційною PKI.
Порівняльний аналіз з централізованою PKI підтвердив переваги запропонованого децентралізованого рішення: відсутність єдиної точки відмови, можливість офлайн-верифікації, нульова вартість реєстрації в тестовій мережі, повний контроль власника над ідентифікатором та стійкість до компрометації.

Таким чином, мету дослідження досягнуто, а всі поставлені завдання виконано. Розроблена децентралізована система управління цифровою ідентичністю може бути використана як основа для створення промислових рішень у сфері безпеки IoT-мереж, а отримані результати можуть бути впроваджені в практичну діяльність підприємств, що використовують розширені IoT-пристрої в корпоративних мережах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aamir M., Sharma S., Grover A. ChaCha20-in-Memory for Side-Channel Resistance in IoT Edge-Node Devices. IEEE Open Journal of Circuits and Systems. 2021. Vol. 2. P. 1-10.
2. Adriaanse P. E. A. A Comparative Study of the TEA, XTEA, PRESENT and Simon lightweight cryptographic schemes : Master's Thesis. Delft : Delft University of Technology, 2021. 21 p.
3. Allen C. The Path to Self-Sovereign Identity [Електронний ресурс] / C. Allen. Life With Alacrity, 2016. - Режим доступу: https://www.lifewithalacrity.com/article/the- path-to-self-soverereign-identity/ (дата звернення: 29.05.2026).
4. Bernal D., Ledesma O., Lamo P., Bermejo J. IOTA-Based Authentication for IoT Devices in Satellite Networks. Computers, Materials & Continua. 2025. Vol. 1. P. 1-39.
5. Bernstein D. J., Duif N., Lange T., Schwabe P., Yang B. High-speed high-security signatures. Journal of Cryptographic Engineering. 2011. Vol. 2. P. 77-89.
6. Bkheet S. A., Agbinya J. I. A Review of Identity Methods of IOT. Advances in Internet of Things. 2021. Vol. 11. P. 153-174.
7. El-Hajj M., Beune P. Lightweight public key infrastructure for the Internet of Things: A systematic literature review. Journal of Information Security and Applications. 2024.
8. Fragoso-Rodriguez U., Laurent-Maknavicius M., Incera-Dieguez J. Federated Identity Architectures. Препринт. 2008. 8 p.
9. Friansa K., Haq I. N., Santi B. M., Kurniadi D., Leksono E., Yuliarto B. Development of Battery Monitoring System in Smart Microgrid Based on Internet of Things (IoT). Procedia Engineering. 2017. Vol. 170. P. 482-487.
10. HKCERT. DigiNotar CA security breach resulting in issuance of fake

certificates [Електронний ресурс] / Hong Kong Computer Emergency Response Team, 2011. - Режим доступу: https://www.hkcert.org/blog/diginotar-ca-security-breach-resulting-in- issuance-of-fake-certificates (дата звернення: 29.05.2026).

11. Fujii H., Aranha D. F. Curve25519 for the Cortex-M4 and beyond. *Progress in Cryptology - LATINCRYPT 2021*. 2021.
12. Goel A., Rahulamathavan Y. A Comparative Survey of Centralised and Decentralised Identity Management Systems: Analysing Scalability, Security, and Feasibility. Future Internet. 2025. Vol. 17. No. 1. P. 1-29.
13. IOTA Foundation. Decentralized Identifiers (DID) [Електронний ресурс] / IOTA Wiki. 2025. - Режим доступу: https://docs.iota.org/developer/iota-identity/explanations/decentralized-identifiers (дата звернення: 25.05.2026).
14. IOTA Foundation. Resolve an IOTA Identity [Електронний ресурс] / IOTA Wiki. 2025. - Режим доступу: https://docs.iota.org/developer/iota-identity/how-tos/decentralized- identifiers/resolve (дата звернення: 25.05.2026).
15. Mahdi M. S., Hassan N. F., Abdul-Majeed G. H. An improved chacha algorithm for securing data on IoT devices. SN Applied Sciences. 2021. Vol. 3. P. 1-9.
16. Prodanović R. I., Vulić I. B. Failure points in the PKI architecture. Vojnotehnički glasnik
/ Military Technical Courier. 2017. Vol. 65. No. 3. P. 771-784.
17. Ramirez-Gordillo T., Macia-Lillo A., Pujol F. A., Garcia-D'Urso N., Azorin-Lopez J., Mora H. Decentralized Identity Management for Internet of Things (IoT) Devices Using IOTA Blockchain Technology. Future Internet. 2025. Vol. 17. No. 1. P. 49.
18. Sedi Nzakuna P., Paciello V., Lay-Ekuakille A., Kuti Lusala A., Dello Iacono S., Pietrosanto A. From IOTA Tangle 2.0 to Rebased: A Comparative Analysis of Decentralization, Scalability, and Suitability for IoT Applications. Sensors.

2025. Vol.

25. No. 11. P. 3408.

19. Sönmez Turan M., McKay K. A., Chang D., Kang J., Kelsey J. Ascon-Based Lightweight Cryptography Standards for Constrained Devices. Gaithersburg, MD : National Institute of Standards and Technology, 2025. (NIST SP 800-232). 52 p.
20. W3C. Decentralized Identifiers (DIDs) v1.0 [Електронний ресурс] / W3C Recommendation, 2022. - Режим доступу: https://www.w3.org/TR/did-1.0/ (дата звернення: 29.05.2026).
21. W3C. Verifiable Credentials Data Model v2.0 [Електронний ресурс] / W3C Recommendation, 2025. - Режим доступу: https://www.w3.org/TR/vc-data-model/ (дата звернення: 29.05.2026).
22. Won J., Singla A., Bertino E., Bollela G. Decentralized public key infrastructure for Internet-of-Things. *MILCOM 2018 - 2018 IEEE Military Communications Conference*. 2018. P. 907-913.
23. ISO/IEC 29167-11:2023. Information technology - Automatic identification and data capture techniques - Part 11: Crypto suite PRESENT-80 security services for air interface communications. Geneva : ISO, 2023.
24. ISO/IEC 29192-1:2012. Information technology. Security techniques. Lightweight cryptography. Part 1: General. Geneva : ISO, 2012. 18 p.
25. Петренко А. І. Криптологія в Інтернеті речей / А. І. Петренко // Сучасний стан та перспективи розвитку інформаційних систем. – 2019. – С. 156-161.
26. Чепель Л. В., Бойко Ю. В. Підхід до безпеки та організації мереж IOT з використанням блокчейн технології / Л. В. Чепель, Ю. В. Бойко // Вісник Вінницького політехнічного інституту. - 2024. - № 4. - С. 133-142.
27. Шматко О. В., Сальніков С. С. Модель децентралізованої системи обміну електричними медичними картками на основі технології блокчейн / О. В. Шматко, С. С. Сальніков // Сучасний стан та перспективи розвитку

інформаційних систем. - 2024. - № 2. - С. 155-162.

28. Сіпко О. М. Децентралізовані IoT-мережі: блокчейн для безпеки та автономності / О. М. Сіпко // Таврійський науковий вісник. Серія: Технічні науки. - 2025. - № 1. - С. 210-215.
29. Явніков Р. Д. Метод ідентифікації пристроїв IoT на базі архітектури цифрових об'єктів : атестаційна робота / Р. Д. Явніков. - Харків : Харківський національний університет радіоелектроніки, 2020. - 77 с.