МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н.Каразіна
Факультет математики і інформатики
Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

магістр

на тему
“Децентралізована система зберігання та обміну медичними даними з пацієнтоорієнтованим керуванням доступом на основі блокчейну Cosmos”

Виконав: студент 6 курсу, групи МФ-61
спеціальність 122 «Комп’ютерні науки»
освітня програма «Теоретична і прикладна інформатика»
Кудрявцев Д. Ю.
Керівник: к. т. н., доцент Меняйлов Є.С.
Рецензент:
_____

Харків – 2026 року

Зміст

Перелік позначень та скорочень

EHR	Electronic Health Records (електронні медичні записи)
EMR	Electronic Medical Records (електронні медичні картки)
PHR	Personal Health Record (персональний медичний запис)
PACS	Picture Archiving and Communication System (система архівування та обміну медичними зображеннями)
HL7	Health Level Seven (стандарт обміну медичними даними)
FHIR	Fast Healthcare Interoperability Resources
GDPR	General Data Protection Regulation (Загальний регламент про захист даних ЄС)
SDK	Software Development Kit
ABCI	Application Blockchain Interface
ADR	Architecture Decision Record
BFT	Byzantine Fault Tolerance
PoS	Proof-of-Stake
IBC	Inter-Blockchain Communication
ICS	Interchain Standards
EVM	Ethereum Virtual Machine
KYC	Know Your Customer
S3	Simple Storage Service (стандарт об'єктного сховища)
CI/CD	Continuous Integration / Continuous Deployment
CLI	Command Line Interface (інтерфейс командного рядка)
TPS	Transactions Per Second (транзакцій на секунду)

Вступ

1.1 Формулювання задачі та обґрунтування актуальності теми

Сучасна охорона здоров’я переживає масштабний перехід від паперового документообігу до електронних медичних записів (Electronic Health Records, EHR). Лікарі вносять діагнози у цифрові системи, результати аналізів зберігаються в базах даних лабораторій, томографічні знімки розміщено у внутрішніх PACS-серверах лікарень. Цей перехід суттєво пришвидшує надання медичної допомоги, але одночасно породжує цілу низку питань, які у паперовому світі взагалі не виникали, бо постало питання володіння даними, цілісності записів та обміну інформацією між закладами.

Розглянемо звичайну ситуацію. Пацієнт роками лікувався у поліклініці за місцем реєстрації. Згодом він переїхав до іншого міста і звертається до нового сімейного лікаря. Фактично у нового лікаря немає жодного способу отримати медичну історію цього пацієнта, окрім як попросити в нього самого паперову виписку або звернутися до старої поліклініки з офіційним запитом, відповідь на який може йти тижнями. Якщо ж пацієнт принесе виписку, у нового лікаря немає жодної можливості перевірити, чи дійсно в ній відображені реальні записи з попередньої системи, чи вона була змінена чи підроблена.

Інша ситуація. Працівник реєстратури або системний адміністратор лікарні має прямий доступ до бази даних, де розміщено медичні записи. Жодних програмних гарантій, що він не може нишком змінити запис у картці пацієнта - поставити чи прибрати діагноз, виправити дату прийому, видалити результати аналізів, фактично не існує. Усі його дії, в кращому випадку, потраплять до логів того ж сервера, який він сам і обслуговує.

Третя ситуація стосується самого пацієнта як власника даних. У більшості країн юридично пацієнт має право на свою медичну

інформацію, але технічно він залежить від лікарні-оператора системи: щоб отримати власні записи, треба написати заяву, щоб дозволити іншому лікарю переглянути ваш діагноз, і тому теж треба пройти через адміністрацію закладу. Ситуація, у якій сам пацієнт надає або відкликає доступ конкретному лікарю безпосередньо, без посередників, у класичних централізованих системах не реалізована майже ніде.

Усі три проблеми мають спільне джерело - централізована архітектура EHR, у якій лікарня одночасно є розпорядником даних, технічним адміністратором системи і єдиним арбітром доступу. Поки усі ці ролі сконцентровані в одному місці, програмними засобами розв’язати наведені проблеми неможливо.

Технологія блокчейну адресує кожну з цих проблем. Незмінний журнал виключає можливість «тихих» правок: будь-яка зміна документа супроводжується публікацією нового хешу, а попередні версії залишаються доступними для перевірки. Криптографічні ключі переводять роль «власника даних» від оператора системи до самого пацієнта, і без його підпису жоден дозвіл на доступ не може бути виданий. Спільний для всіх лікарень реєстр прибирає необхідність у централізованому посереднику для обміну даними між закладами: коли пацієнт переходить до нового лікаря, йому достатньо видати йому дозвіл криптографічно, а лікар зможе одразу запросити документи у попередньої лікарні, без очікування на офіційну відповідь.

Однак переносити медичні записи на блокчейн напряму було б помилкою. По-перше, це порушило б принцип мінімізації обробки персональних даних, закріплений у GDPR та Законі України «Про захист персональних даних». По-друге, це зробило б неможливим виконання права на забуття: блокчейн принципово незмінний. По-третє, обсяги медичних даних (особливо знімків) зробили б таку мережу непрацездатною. Тому актуальною інженерною задачею є не просто

«винести EHR у блокчейн», а спроєктувати гібридну архітектуру, у якій блокчейн виступає шаром цілісності та контролю доступу, а самі документи залишаються у сховищах лікарень, але з прив'язкою до on-chain хешів, що робить будь-яке несанкціоноване редагування одразу виявленим.

Мета роботи:

1) Дослідити існуючі підходи до побудови blockchain-based EHR систем та проаналізувати їхні обмеження.
2) Дослідити Cosmos SDK як платформу побудови прикладно-специфічних блокчейнів та обґрунтувати її переваги порівняно з універсальними смарт-контрактними платформами для задач охорони здоров'я.
3) Спроєктувати гібридну on-chain/off-chain архітектуру системи з чітким розподілом зон відповідальності - на блокчейні зберігаються реєстри суб'єктів, дозволи доступу та хеші документів, у сервісах лікарень - самі медичні документи.
4) Розробити модель пацієнтоорієнтованого керування доступом з гранулярністю на рівні прав (CREATE, READ, UPDATE) та області дії (окремий документ або вся медична картка в межах закладу).
5) Реалізувати прикладно-специфічний блокчейн на базі Cosmos SDK з кастомними модулями для реєстрації лікарень, лікарів і пацієнтів, управління дозволами та зберігання хешів документів.
6) Реалізувати бекенд-сервіс лікарні, що верифікує підписи запитів лікарів за стандартом ADR-036, звіряє права доступу зі станом блокчейну та забезпечує публікацію хешів документів перед їх збереженням у off-chain сховище.
7) Провести функціональне, інтеграційне та базове тестування безпеки розробленого прототипу, оцінити продуктивність та проаналізувати стійкість до типових атак.

8) Розглянути перспективи подальшого розвитку системи, включно з міжмережевим обміном через ІВС та інтеграцією зі стандартами медичного обміну даними HL7 FHIR.

1.2 Огляд існуючих рішень для управління медичними записами на основі блокчейну

Ідея застосування блокчейн-технологій у сфері охорони здоров'я не нова - за останнє десятиліття було запропоновано кілька дослідницьких прототипів та комерційних платформ. Розглянемо найбільш відомі з них, щоб виявити їхні сильні та слабкі сторони.

1. **MedRec**[1]: дослідницький проєкт Массачусетського технологічного інституту (MIT Media Lab), один з перших серйозних прототипів blockchain-based EHR. Побудований на платформі Ethereum, використовує смарт-контракти для управління посиланнями на медичні записи та правами доступу. Система оперує трьома типами контрактів: Registrar Contract (реєстрація користувачів), Patient-Provider Relationship Contract (зв'язки пацієнт-лікар) та Summary Contract (агрегація посилань на записи). Основні обмеження: залежність від публічної мережі Ethereum з відповідною вартістю транзакцій, обмежена гнучкість моделі дозволів, відсутність підтримки міжзакладного обміну.
2. **Medicalchain**[2]: комерційна платформа на базі Hyperledger Fabric з додатковим публічним токеном. Орієнтована на надання пацієнтам контролю над своїми записами та можливості монетизації даних для медичних досліджень. Архітектура передбачає двошарову модель - приватний блокчейн для медичних даних і публічний для токеноміки. Серед обмежень - закритість значної частини коду, складність розгортання Hyperledger Fabric з його CA-серверами та політиками каналів, відсутність відкритих публікацій про реальні впровадження.
3. **MyHealthMyData**[3]: європейський дослідницький проєкт, фінансований у межах програми Horizon 2020. Використовує федеративну архітектуру з блокчейном для аудиту дій з даними.

Основний акцент на узгодженості з GDPR. Проєкт продемонстрував працездатність концепції, але залишився на рівні дослідницького прототипу і не отримав комерційного продовження.

4. **OmniPHR**[4]: академічна архітектура персональних медичних записів, що використовує концепцію розподіленого зберігання та блокчейн для координації доступу. Цікава своєю формальною моделлю, але реалізація обмежена дослідницьким прототипом без реальних впроваджень.
5. **Інші проєкти на Hyperledger Fabric**: низка дослідницьких робіт пропонують різні варіації архітектур на базі Fabric (наприклад, Healthchain, BlocHIE). Усі вони успадковують переваги і недоліки самої платформи Fabric: продуктивність на рівні корпоративних СУБД, але складність розгортання, специфічна модель endorsement policies та обмежена можливість міжмережевої взаємодії.

Аналіз існуючих рішень дозволяє виділити кілька повторюваних обмежень. По-перше, переважна більшість використовує універсальні смарт-контрактні платформи (Ethereum, Hyperledger Fabric), що накладає накладні витрати віртуальної машини та обмежує гнучкість моделі стану. По-друге, моделі дозволів у наявних рішеннях, як правило, бінарні (доступ є або доступу немає) і не підтримують гранулярності за типом операції (CREATE / READ / UPDATE) та за областю дії (документ / картка). По-третє, питання міжмережевої взаємодії між блокчейнами різних медичних консорціумів практично не розглянуто - кожен проєкт існує як ізольована мережа.

Перелічені обмеження залишають місце для нового підходу, який поєднав би переваги прикладно-специфічного блокчейну (відсутність накладних витрат VM, повний контроль над моделлю стану) з гнучкою моделлю дозволів та можливістю майбутньої інтеграції в єдину екосистему медичних мереж через стандартизований протокол міжмережевого обміну.

1.3 Обґрунтування використання Cosmos SDK та порівняння з альтернативними платформами

Cosmos SDK[5] - це модульний фреймворк для побудови прикладно-специфічних блокчейнів (application-specific blockchains) мовою Go, розроблений спільнотою Cosmos. На відміну від універсальних смарт-контрактних платформ, які пропонують єдине загальне середовище виконання для довільних застосунків, Cosmos SDK дозволяє побудувати блокчейн, вся логіка якого підпорядковується конкретній предметній області, у нашому випадку, управлінню медичними записами.

Cosmos SDK має модульну структуру та містить набір стандартних модулів, необхідних для побудови будь-якої блокчейн-мережі, такі як Auth (облікові записи користувачів), Bank (баланси та переміщення токенів), Staking (управління валідаторами), Gov (внутрішньомережеве голосування), Authz (делегування повноважень), Upgrade (координовані оновлення мережі) тощо. Розробники також мають можливість створювати власні модулі за стандартизованими патернами Cosmos SDK, що значно прискорює розробку та забезпечує однорідну архітектуру коду.

Cosmos SDK дозволяє використовувати різні алгоритми досягнення консенсусу через стандартизований інтерфейс ABCI (Application Blockchain Interface). Зазвичай використовується CometBFT[6] (раніше - Tendermint Core), що реалізує BFT-консенсус з миттєвою фінальністю блоків та стійкістю до 1/3 компрометованих валідаторів. Така модель особливо підходить для permissioned-мереж медичних консорціумів, де набір валідаторів визначається явно, а вимоги до часу фінальності транзакцій є критичними.

Для проєкту управління медичними записами архітектурні особливості Cosmos SDK дають кілька принципових переваг:

- Відсутність віртуальної машини усуває накладні витрати на інтерпретацію смарт-контрактного байт-коду, уся логіка виконується як нативний скомпільований Go-код.
- Прямий доступ до структури стану через інтерфейс Keeper дозволяє ефективно реалізувати складні структури даних (наприклад, версіонований журнал документів), що було б обтяжливо через смарт-контракти.
- Явна модель governance дає природний механізм керування permissioned-мережею медичного консорціуму - додавання нової лікарні до набору валідаторів проходить як стандартна governance-процедура без необхідності будувати окремий рівень управління.

Після аналізу інших платформ, на яких можна було б побудувати систему управління медичними записами, було сформовано порівняльну таблицю.

Критерій порівняння	**Cosmos SDK**	**Ethereum**	**Hyperledger Fabric**
Тип архітектури	Прикладно-специфічний блокчейн	Універсальна смарт-контрактна платформа	Permissioned-блокчейн з модульною архітектурою
Мова реалізації бізнес-логіки	Go	Solidity, Vyper	Go, Java, JavaScript
Накладні витрати виконання	Відсутні (нативний код)	EVM-інтерпретація	Контейнеризація chaincode

Алгоритм консенсусу	CometBFT (BFT, миттєва фінальність)	PoS (Casper FFG, відносна фінальність)	Raft або PBFT (на рівні ordering service)
Модель Permissioned	Через governance + явний набір валідаторів	Складно (потребує приватного форку)	Нативна
Час фінальності блоку	1–5 с	~12 хв (12 епох)	1–3 с
Гнучкість моделі стану	Висока (прямий KVStore доступ)	Обмежена смарт-контрактом	Обмежена ledger world state
Міжмережева взаємодія	Нативна через IBC	Через мости (з компромісами безпеки)	Через окремі рішення (Fabric Interop)
Складність розгортання вузла	Один бінарник	Один клієнт (Geth/Erigon)	Багатокомпонентна (peer + orderer + CA)

Таблиця 1.1 — Порівняльна таблиця платформ для побудови блокчейн-системи управління медичними записами

Із порівняння видно, що Cosmos SDK переважає Ethereum за всіма критеріями, важливими для медичного сценарію: відсутність накладних витрат VM, миттєва фінальність, нативний permissioned-режим, гнучкість моделі стану. Hyperledger Fabric лишається сильним конкурентом і теоретично міг би бути обраний для даної задачі, але має кілька суттєвих практичних мінусів: значно складніша архітектура розгортання (peer-вузли, orderer-сервіси, окремі сертифікаційні центри для кожної організації, політики ендорсменту для кожного chaincode), помітно

складніший процес розробки та тестування, відсутність нативної підтримки міжмережевої взаємодії на рівні самого протоколу.

Окремо варто зазначити особливість екосистеми Cosmos. Тоді як Ethereum та Hyperledger Fabric формують відносно ізольовані мережі, всю екосистему cosmos-based мереж можна розглядати як єдиний організм , що складається з мереж, спеціалізованих під певні предметні області та поєднаних через протокол IBC [7] (Inter-Blockchain Communication). Цей протокол додається як окремий модуль і дозволяє передавати будь-які дані між мережами та навіть виконувати транзакції в одній мережі з результатом, доступним в іншій. У контексті даної роботи це відкриває перспективу майбутньої інтеграції розробленого блокчейну з іншими медичними мережами (наприклад, з мережею страхових компаній або з мережею державного реєстру медичних послуг) без необхідності перебудови архітектури, виключно через додавання IBC-модуля.

Таким чином, Cosmos SDK обрано як платформу реалізації з огляду на сукупність технічних переваг (відсутність накладних витрат VM, миттєва фінальність, гнучкість моделі стану), архітектурної відповідності задачі (нативний permissioned-режим, прямий доступ до KVStore) та перспектив подальшого розвитку (міжмережевий обмін через IBC).

Основна частина

2.1 Постановка задачі

Розробка децентралізованої системи зберігання та обміну медичними даними з пацієнтоорієнтованим керуванням доступом передбачає створення двох взаємопов’язаних компонентів: прикладно-специфічного блокчейну на базі Cosmos SDK, що виступає єдиним джерелом істини для реєстрів суб’єктів, дозволів доступу та хешів документів, та бекенд-сервісу лікарні, який зберігає самі медичні документи у власному off-chain сховищі та обслуговує підписані запити лікарів з обов’язковою перевіркою прав доступу зі станом блокчейну.

Під час проєктування системи треба буде врахувати кілька принципових архітектурних обмежень. По-перше, медичні документи з огляду на принцип мінімізації обробки персональних даних (закріплений у GDPR та ЗУ «Про захист персональних даних») та обмеження права на забуття не можуть зберігатися on-chain - лише їхні криптографічні хеші. По-друге, уся бізнес-логіка модулів блокчейну має бути строго детермінованою, оскільки будь-яке використання системного часу, випадкових чисел або неупорядкованих колекцій призводить до розбіжності стану між валідаторами та зупинки мережі. По-третє, протокол автентифікації запитів між лікарем та сервером лікарні має бути стійким до replay-атак та cross-chain replay, при цьому сумісним з існуючою криптографічною інфраструктурою Cosmos SDK.

Для коректної роботи з блокчейном та забезпечення сумісності клієнтських інструментів з усією екосистемою Cosmos треба буде використовувати стандартний протокол ADR-036[8] (Arbitrary Message Signature Specification) для підпису off-chain запитів, secp256k1 для криптографії облікових записів та формат Bech32 для адрес. Це дозволить використовувати стандартні Cosmos-гаманці (Keplr, Cosmostation, Ledger)

для зберігання ключів пацієнтів та лікарів без необхідності розробляти власний клієнтський інструментарій.

Блокчейн-шар має забезпечувати такі можливості:

- реєстрація лікарень адміністратором мережі через governance-механізм;
- реєстрація лікарів з обов'язковим закріпленням за конкретною лікарнею;
- реєстрація пацієнтів як самосуверенних облікових записів;
- видача пацієнтом дозволів конкретному лікарю з гранулярністю на рівні прав (CREATE, READ, UPDATE) та області дії (окремий документ або вся медична картка в межах закладу);
- відкликання дозволів пацієнтом у будь-який момент;
- анкорування хешів медичних документів з прив'язкою до автора, часу та версії;
- збереження повної історії версій документа без можливості перезапису попередніх;
- публікація блокчейн-подій для зовнішніх споживачів через WebSocket CometBFT;
- запит поточного стану через gRPC та REST API.

Бекенд-сервіс лікарні має забезпечувати такі можливості:

- REST/gRPC API для створення, читання та оновлення медичних документів;
- верифікація підписаних запитів лікарів за стандартом ADR-036 з захистом від replay-атак;
- звірка прав доступу зі станом модуля x/access через gRPC до повнонoди блокчейну;
- синхронне анкорування хешів документів on-chain перед їх збереженням у off-chain сховище;

- off-chain сховище медичних документів з підтримкою як структурованих клінічних даних (PostgreSQL з JSONB), так і бінарних артефактів (об’єктне сховище MinIO);
- підписка на блокчейн-події через WebSocket CometBFT для синхронізації локального індексу;
- локальний журнал доступу до документів для внутрішнього аудиту лікарні.

Детальний опис алгоритмів роботи блокчейн-модулів та протоколу автентифікації запитів буде описаний у наступних підрозділах.

2.2 Опис алгоритмів та структури блокчейн-модулів

Дослідження архітектури Cosmos SDK та аналіз існуючих cosmos-based мереж, що працюють з реєстрами суб'єктів та системами контролю доступу, дозволили сформулювати декомпозицію бізнес-логіки на п'ять власних модулів. Кожен модуль інкапсулює окрему предметну область, має чіткі межі відповідальності та взаємодіє з іншими модулями виключно через публічні методи Keeper-ів.

Нижче наведено опис ключових модулів та алгоритмів їх роботи:

1) **Модуль x/hospital**

Модуль зберігає реєстр медичних закладів, що входять до мережі. Кожна лікарня зареєстрована адміністратором мережі (authority address у термінах Cosmos SDK v0.50) і характеризується унікальним ідентифікатором, Bech32-адресою свого бекенд-сервісу, людино-читаним найменуванням та статусом (активна / деактивована). Реєстрація нової лікарні відбувається через стандартний governance-механізм - пропозиція виноситься на голосування існуючих валідаторів і приймається при досягненні кворуму. Це забезпечує природну модель управління permissioned-мережею медичного консорціуму без необхідності будувати окремий рівень адміністрування.

Модуль використовує транзакції MsgRegisterHospital, MsgUpdateHospitalEndpoint, MsgDeactivateHospital та запити QueryHospital(id) та QueryHospitalList. Усі транзакції з модифікацією стану доступні лише authority-address.

2) Модуль x/doctor

Модуль зберігає реєстр лікарів із обов'язковим закріпленням за конкретною лікарнею. Кожен лікар має окремий Cosmos-обліковий запис із власним приватним ключем, що дозволяє йому самостійно підписувати запити до бекенду лікарні без посередництва закладу. Реєстрація лікаря

виконується від імені адміністратора відповідної лікарні (вказаного в реєстрі x/hospital) через транзакцію MsgRegisterDoctor.

Структура запису включає Bech32-адресу облікового запису лікаря, ідентифікатор лікарні-роботодавця, посаду (для людино-читаного UI) та статус (активний / деактивований). При спробі лікаря виконати дію (наприклад, анкорувати документ) інші модулі звертаються до x/doctor для перевірки активності лікаря та його прив’язки до конкретної лікарні.

3) Модуль x/patient

Реєстр пацієнтів спроєктований за принципом self-sovereign identity - обліковий запис пацієнта створюється самим користувачем шляхом самопідписаної транзакції MsgRegisterPatient, без потреби у схваленні з боку будь-якого закладу. Це забезпечує реалізацію принципу інформаційного самовизначення пацієнта, який є фундаментальним для всієї системи.

Окремо передбачена операція MsgConfirmPatientCard - підписана лікарнею транзакція, що пов’язує блокчейн-адресу пацієнта з конкретним записом у медичній картці закладу. Така подвійна реєстрація розв'язує практичну задачу: пацієнт міг лікуватися в лікарні до того, як отримав блокчейн-адресу, і потрібен спосіб надійно прив’язати накопичену медичну історію до його нового цифрового аналога.

4) Модуль x/access

Це ключовий модуль системи, у якому реалізовано пацієнтоорієнтовану модель керування доступом. Дозвіл (Grant) у системі представлений кортежем (patient, doctor, scope, rights, expires_at), де:

- patient - Bech32-адреса пацієнта-власника медичних даних;
- doctor - Bech32-адреса лікаря, якому видається дозвіл;
- scope - область дії дозволу: або конкретний document_id, або (hospital_id, "all") для надання доступу до всієї медичної картки в межах конкретного закладу;

- rights - бітмаска прав, що поєднує CREATE (створення нових документів), READ (читання) та UPDATE(оновлення існуючих) - підтримуються будь-які комбінації;
- expires_at - абсолютний час спливу дозволу (у форматі block time), після якого дозвіл вважається відсутнім без необхідності окремого відкликання.

Видача дозволу виконується самим пацієнтом через MsgGrantAccess, відкликання - через MsgRevokeAccess. У жодному разі ані лікарня, ані лікар не можуть видати дозвіл на доступ до даних пацієнта без його явного підпису. Перевірка прав доступу зведена до запиту QueryAccess(patient, doctor, scope), що повертає поточний дозвіл з урахуванням строку дії; за його відсутності або після спливу expires_at запит повертає порожнє значення.

Окремо реалізовано circuit breaker - параметр paused у конфігурації модуля, що може бути встановлений через governance-голосування у разі виявлення критичної вразливості. При активному paused усі повідомлення з модифікацією стану повертають помилку, що дозволяє екстрено зупинити операції на час розслідування інциденту.

5) Модуль x/document

Модуль зберігає хеші медичних документів з прив’язкою до автора, лікарні, часу та версії. Самі документи on-chain не зберігаються - лише їхні SHA-256 хеші, обчислені за канонічною формою (для JSON - з відсортованими ключами та без зайвих пробілів; для бінарних даних - за байтовим вмістом). Така адресація за хешем (content-addressable storage) перетворює саму модель зберігання на криптографічно перевірювану: будь-яка несанкціонована зміна документа в off-chain сховищі негайно виявляється при наступному запиті, оскільки обчислений хеш не співпаде зі збереженим on-chain.

Структура запису включає унікальний document_id (генерується послідовно), Bech32-адресу пацієнта-власника, ідентифікатор лікарні-розпорядника, актуальний номер версії та хеш кожної версії з посиланням на її автора (лікаря) та час публікації. Принциповою особливістю є **імутабельність попередніх версій** - нова версія документа додається через MsgUpdateDocument як окремий запис у журналі версій, без перезапису або видалення попередніх. Таким чином, повна історія змін документа залишається доступною для аудиту, а відповідь на запит QueryDocumentHistory(id) повертає послідовність усіх версій з відповідними хешами та підписами авторів.

Перед виконанням MsgAnchorDocument або MsgUpdateDocument модуль через міжмодульний виклик звертається до x/access для перевірки наявності у автора-лікаря відповідного права (CREATE для нового документа, UPDATE для нової версії), а до x/doctor - для перевірки активності та прив’язки лікаря до тієї ж лікарні, що вказана в документі.

Інваріанти модулів

На рівні всієї блокчейн-логіки зафіксовано набір інваріантів, що перевіряються через стандартний механізм Invariants registrar Cosmos SDK на кожному блоці:

- лікар може анкорувати документи лише для тієї лікарні, у якій він зареєстрований;
- дозвіл з expires_at <= ctx.BlockTime() не повинен авторизувати жодну дію;
- після запису хешу версії документа цей хеш є незмінним - нові версії додаються, попередні не редагуються;
- тільки сам пацієнт (або делегат через x/authz) може видавати чи відкликати дозволи на свої дані.

Порушення будь-якого з інваріантів зупиняє виконання транзакції з відповідною помилкою.

Враховуючи описану структуру модулів, можемо сформулювати алгоритм виконання типового сценарію - анкорування нового документа лікарем:

1) Лікар через CLI або клієнтський застосунок формує транзакцію MsgAnchorDocument з полями (doctor, patient, hospital_id, content_hash), підписує її своїм приватним ключем та відправляє на повнонoду мережі.
2) Транзакція потрапляє в мемпул, поширюється усім валідаторам мережі та обирається пропоузером поточного блоку для включення.
3) Cosmos SDK BaseApp роутить виконання транзакції до Msg-обробника модуля x/document.
4) У функції ValidateBasic() перевіряється формальна валідність полів транзакції (формат адрес, довжина хешу, ненульові значення).
5) Msg-обробник через Keeper модуля x/doctor перевіряє, що відправник зареєстрований як активний лікар і прив’язаний до зазначеної лікарні.
6) Через Keeper модуля x/access перевіряється наявність у лікаря дозволу CREATE від пацієнта на створення документів у межах вказаної лікарні (або на конкретний документ, якщо scope був попередньо налаштований). Перевірка також враховує expires_at дозволу.
7) У разі успішних перевірок у KVStore модуля x/document записується новий запис з присвоєнням наступного document_id, версією 1, хешем та автором.
8) Виконується emit блокчейн-події EventDocumentAnchored з усіма ключовими полями (document_id, patient, doctor, hospital, hash, version), що буде отримана бекенд-сервісом лікарні через WebSocket CometBFT.

9) Транзакція успішно завершується, Bekend-сервіс лікарні отримує подію та зберігає сам документ у власне off-chain сховище з прив’язкою до отриманого document_id.

Алгоритм оновлення документа (MsgUpdateDocument) працює аналогічно, але з перевіркою права UPDATE замість CREATE та зі створенням нової версії в журналі замість нового запису.

2.3 Огляд протоколу автентифікації запитів за стандартом ADR-036

Між лікарем та бекенд-сервісом лікарні існує окремий канал взаємодії, що не пов’язаний з блокчейн-транзакціями - лікар надсилає HTTP/gRPC-запити для отримання, створення та оновлення медичних документів. Ці запити повинні автентифікуватися криптографічно: сервер має бути впевнений, що запит дійсно надійшов від конкретного лікаря, а не від зловмисника, який перехопив трафік або викрав сесію. Водночас використовувати окремі облікові записи (паролі, JWT-токени) було б архітектурно неправильно - це створило б додатковий рівень довіри, паралельний до блокчейн-системи, з усіма наслідками у вигляді витоків і компрометацій.

Для розв’язання цієї задачі в системі використано стандарт **ADR-036** (Arbitrary Message Signature Specification) - офіційний механізм Cosmos-екосистеми для підпису довільних повідомлень тим самим приватним ключем, яким підписуються блокчейн-транзакції. Це дозволяє лікарю використовувати свій єдиний Cosmos-обліковий запис як для on-chain взаємодій, так і для автентифікації перед бекендом лікарні, без необхідності керувати декількома наборами облікових даних.

Протокол автентифікації побудовано наступним чином. Перед кожним значущим запитом до бекенду лікар формує structured payload з такими полями:

- chain_id - ідентифікатор блокчейн-мережі (наприклад, medchain-mainnet-1); прив’язує підпис до контексту конкретної мережі та запобігає cross-chain replay;
- sequence - монотонно зростаючий лічильник запитів від конкретного лікаря, унікальний для кожного запиту; запобігає replay-атакам через повторне використання раніше перехопленого підпису;

- operation - тип операції (READ_DOCUMENT, CREATE_DOCUMENT, UPDATE_DOCUMENT); явне зазначення операції в підписаному payload-і запобігає підробці типу запиту шляхом маніпуляції HTTP-методом;
- document_id (для READ/UPDATE) або (patient, hospital_id) для CREATE - ідентифікація цільового ресурсу; гарантує, що підпис, отриманий для одного документа, не може бути використаний для доступу до іншого;
- issued_at - час формування запиту (у форматі Unix timestamp); використовується сервером для перевірки staleness - запити, видані більше ніж 5 хвилин тому, відхиляються незалежно від коректності підпису.

Сформований payload серіалізується в канонічну JSON-форму (відсортовані ключі, без зайвих пробілів), обертається в стандартну ADR-036-обгортку (StdSignDoc з порожнім chain-id для самого ADR-036, але з полем chain_id всередині payload-у) та підписується приватним ключем лікаря через secp256k1.

Перевірка запиту на стороні сервера лікарні відбувається в кілька етапів:

1) із заголовків запиту витягується підпис, публічний ключ та переданий serialized payload;
2) перевіряється відповідність публічного ключа Bech32-адресі лікаря, вказаному в payload-і;
3) через стандартні бібліотеки Cosmos SDK (crypto/keyring) перевіряється криптографічна валідність підпису для даного payload-у та публічного ключа;
4) перевіряється рівність chain_id з payload-у налаштованому ідентифікатору мережі, невідповідність відхиляється з 401 Unauthorized;

5) перевіряється свіжість запиту: |issued_at - current_block_time| повинно бути менше 5 хвилин;
6) із локальної таблиці doctor_sequences витягується останній отриманий sequence для цього лікаря, новий запит повинен мати sequence > last_seen. У разі рівності або меншого значення запит відхиляється як replay-атака. Лічильник оновлюється атомарно лише після повністю успішної автентифікації;
7) через gRPC до повної ноди блокчейну виконується запит QueryDoctor(addr) для перевірки, що лікар активний та прив'язаний до даної лікарні;
8) через gRPC виконується запит QueryAccess(patient=document.patient, doctor=signer, scope=document.id) для перевірки наявності відповідного права (READ для GET-запиту, UPDATE для PUT-запиту тощо);
9) у разі успіху всіх перевірок сервер виконує запит (зчитує документ зі сховища, обчислює хеш та звіряє з on-chain хешем для запитів на читання, формує транзакцію MsgAnchorDocument для запитів на створення).

Окремо опрацьовано граничний випадок порівняння хешу при читанні. Перед поверненням документа лікарю сервер обчислює SHA-256 від збереженого змісту та порівнює з хешем останньої версії, отриманим із модуля x/document. У разі невідповідності повертається 409 Conflict із повідомленням про можливе порушення цілісності off-chain сховища та надсилається сповіщення в систему моніторингу. Це фундаментальна гарантія системи, бо навіть якщо адміністратор лікарні з прямим доступом до бази даних модифікує запис, наступне читання негайно виявить розбіжність з on-chain хешем.

Така архітектура автентифікації дає одночасно три ключові властивості. По-перше, формується єдиний криптографічний контекст:

лікар використовує той самий ключ для блокчейн-транзакцій (видача/відкликання дозволів від його імені, якщо він також є пацієнтом) та для автентифікації перед бекендом, без додаткових облікових даних. По-друге, система не потребує серверного стану сесії: сервер не зберігає токени, не керує таймаутами сесій, адже кожен запит є повністю самодостатнім із криптографічної точки зору. По-третє, досягається стійкість до низки класичних атак: replay через sequence-counter, cross-chain replay через chain_id, підміна типу запиту через явне operation-поле, staleness-атаки через timestamp.

2.4 Деплой і тестування системи

Наступними кроками реалізації системи є:

1) збірка кастомних модулів x/hospital, x/doctor, x/patient, x/access, x/document у складі стандартного Cosmos SDK-додатку та їх реєстрація в module.go;
2) ініціалізація локальної мережі з трьома валідаторами для перевірки досягнення консенсусу та правильності формування блоків;
3) збірка та запуск бекенд-сервісу лікарні з підключенням до локальної мережі;
4) створення тестового набору даних (дві лікарні, троє лікарів, троє пацієнтів) через CLI-команди мережі;
5) виконання кінцевих сценаріїв: первинного прийому, передачі пацієнта іншому спеціалісту, відкликання дозволу;
6) цілеспрямоване тестування механізмів захисту: спроби несанкціонованого доступу, replay-атак, підміни даних в off-chain сховищі.

Розгортання локальної мережі здійснено за допомогою Docker Compose конфігурації, що піднімає одночасно три валідатори блокчейну, два бекенд-сервіси лікарень (по одному на лікарню), два інстанси PostgreSQL та два інстанси MinIO. Конфігурація запускається однією командою docker compose up, що забезпечує повну відтворюваність експериментів на будь-якій машині з Docker.

Рисунок 2.1 — Логи валідаторів мережі під час досягнення консенсусу та формування перших блоків

З логів запуску мережі видно, що три валідатори успішно встановили з'єднання, обмінялися початковими даними про набір валідаторів та через CometBFT почали створювати блоки з інтервалом ~5 секунд. Інтервал блоків можна налаштовувати через параметр consensus.timeout_commit у конфігурації CometBFT. Для продуктивного середовища зазвичай доцільно використовувати значення порядку 1–2 секунд, тоді як для локального тестування зручнішим є інтервал близько 5 секунд, оскільки він полегшує спостереження за процесом роботи мережі.

Реєстрацію кожної з тестових лікарень виконано через стандартний governance-механізм Cosmos SDK: створено пропозицію типу MsgRegisterHospital, проведено голосування валідаторів та після проходження кворуму пропозиція автоматично виконана. Адреси сервісів лікарень зафіксовано у відповідному реєстрі модуля x/hospital.

```
==> Waiting for chain to produce blocks...
==> Chain is live at height 3
==> Governance address: medchain10d07y265gmmuvt4z0w9aw880jnsr700jrm43g9
==> Submitting proposal: Register City General Hospital
gas estimate: 240888
==> Proposal tx: EC0E845BA8C5365C9B34DDAC0EBF2BBF6EE0D31F4B2CE8AFB0027448C1955CD2 — waiting for indexing...
==> Proposal ID: 1
gas estimate: 56334
==>   validator1 voted yes
gas estimate: 56487
==>   validator2 voted yes
gas estimate: 56487
==>   validator3 voted yes
==> Waiting for voting period to end (≈20 s)...
==> Proposal 1 status: PROPOSAL_STATUS_PASSED
==> Submitting proposal: Register North Medical Center
gas estimate: 240658
==> Proposal tx: F3423D0A029BD1EF2A394AB37E45D3F1A3ECBCA631F570FC250ED87DE6DFB318 — waiting for indexing...
==> Proposal ID: 2
gas estimate: 56334
==>   validator1 voted yes
gas estimate: 56487
==>   validator2 voted yes
gas estimate: 56487
==>   validator3 voted yes
==> Waiting for voting period to end (≈20 s)...
==> Proposal 2 status: PROPOSAL_STATUS_PASSED
==>
==> Hospitals registered:
{
  "id": "1",
  "name": "City General Hospital",
  "admin": "medchain1d3dsm49ugevas8nr5nhu5dvs26v6xqj309rcj0",
  "active": true
}
{
  "id": "2",
  "name": "North Medical Center",
  "admin": "medchain1tv4yyg2nu67rnfp325qvue433dw03z724wj9x3",
  "active": true
}
```

Рисунок 2.2 — Результати реєстрації двох лікарень через governance-голосування

```
{
  "addr": "medchain1tgdumqwd89lppgxryvuaukxyc3nutsgwml8kce",
  "confirmed": true,
  "confirming_hospital_id": "2"
}
{
  "addr": "medchain1cnjawhklqkckayuq7l98m4vp7eznu6da568r05",
  "confirmed": true,
  "confirming_hospital_id": "2"
}
==>
==> Access grants:
{
  "patient": "medchain132gnfxsl2n5rallcnnvhlcymauw3lvryusrv8s",
  "doctor": "medchain1r8n0rm9z47tepgvq04gp45lqv9y8lwgrnaznzg",
  "scope": "hospital:1",
  "rights": 7,
  "revoked": null
}
{
  "patient": "medchain1tgdumqwd89lppgxryvuaukxyc3nutsgwml8kce",
  "doctor": "medchain1zt823rmujkqrje8yse4wfgzcnzf4txprq0j8ae",
  "scope": "hospital:2",
  "rights": 7,
  "revoked": null
}
```

Рисунок 2.3 — Результат виконання MsgGrantAccess пацієнтом

На рисунку показано результат успішного виконання транзакції видачі дозволу: пацієнт medchain1...alice видав лікарю medchain1...bob дозвіл READ | UPDATE на конкретний документ зі строком дії 24 години. Видно емітовану подію EventAccessGranted з усіма полями кортежу дозволу. Витрачений газ - близько 60 000 одиниць, що відповідає типовим показникам для повідомлень з невеликою кількістю записів у KVStore.

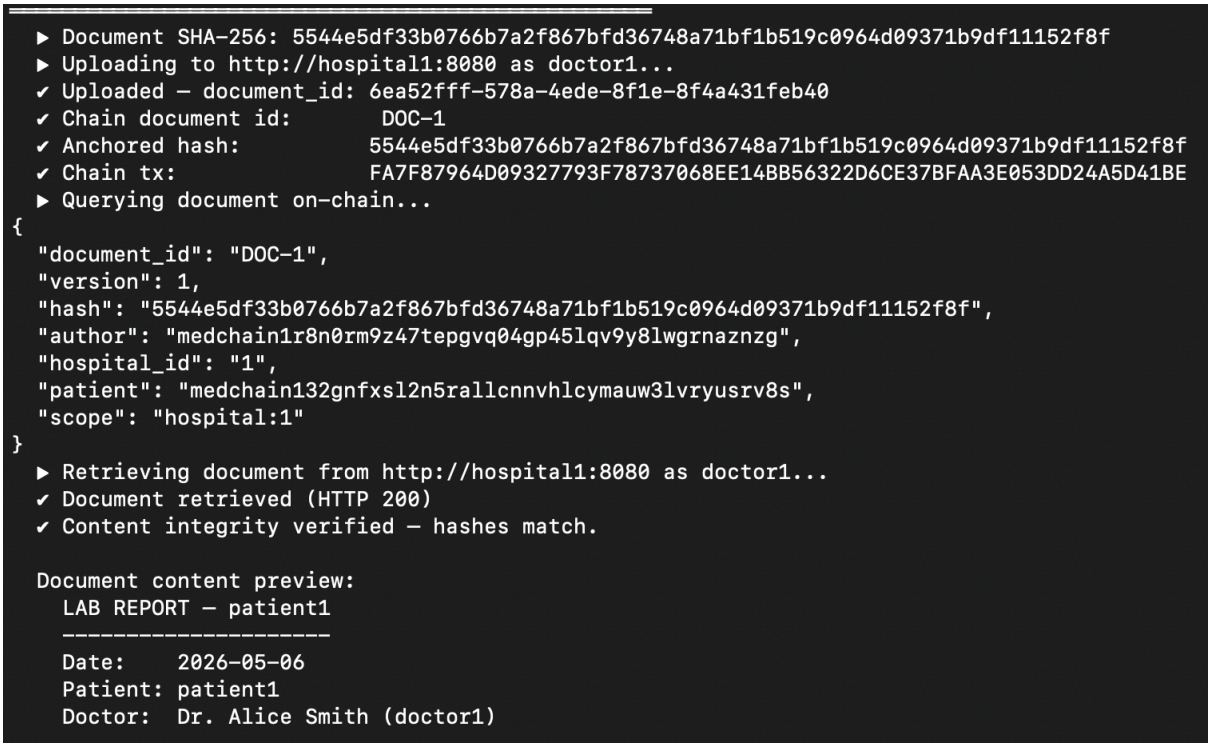

Рисунок 2.4 — Результат виконання MsgAnchorDocument лікарем

Тут представлено результат анкорування нового документа: лікар medchain1...bob опублікував у мережі хеш документа для пацієнта medchain1...alice. До виконання MsgAnchorDocument модуль x/document зробив міжмодульний виклик до x/access для перевірки наявності права CREATE, та до x/doctor для перевірки активності лікаря. Усі перевірки пройшли успішно, документ збережено у KVStore з присвоєним document_id = 1 та версією 1.

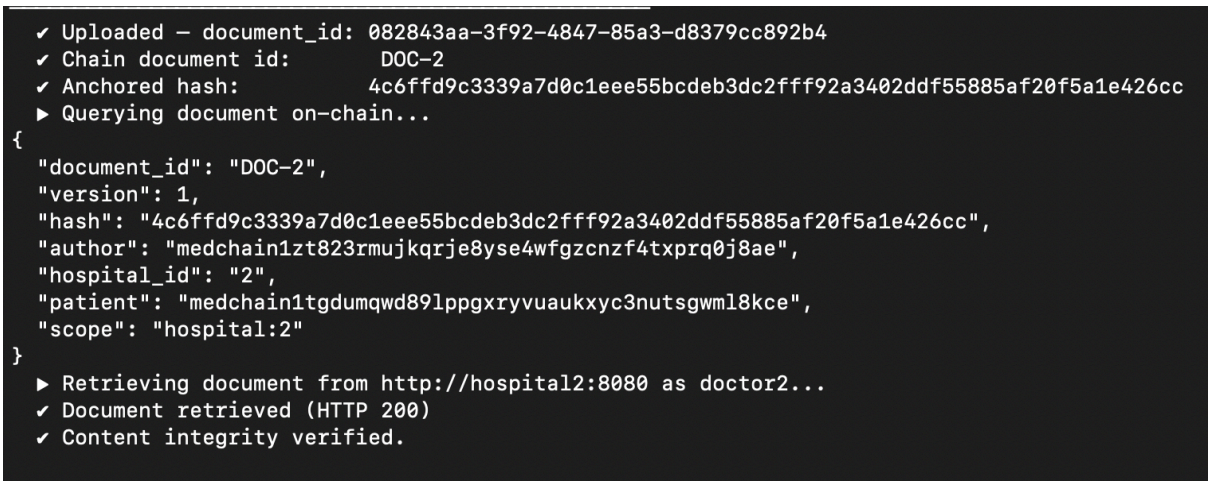

Рисунок 2.5 — Перегляд документа через бекенд-сервіс лікарні з ADR-036-підписаним запитом

Лікар через клієнтський інструмент сформував ADR-036-підписаний запит на читання документа, надіслав його до бекенду лікарні. Сервер послідовно виконав усі етапи перевірки: верифікація підпису, перевірка свіжості, перевірка sequence, запит до блокчейну для перевірки прав, обчислення SHA-256 від збереженого змісту, порівняння з on-chain хешем - і повернув документ. Загальний час обробки склав близько 180 мс, з яких приблизно 80 мс - gRPC-запит до повної ноди.

Тестування механізмів захисту проведено за такими сценаріями:

- **Replay-атака.** Перехоплено ADR-036-підписаний запит лікаря і повторно надіслано через 30 секунд. Сервер коректно ідентифікував повторне використання sequence та повернув 409 Conflict без виконання операції.
- **Запит від лікаря з відкликаним дозволом.** Пацієнт виконав MsgRevokeAccess, після чого лікар спробував отримати документ, який раніше був йому доступний. Бекенд лікарні через gRPC-запит QueryAccess отримав від блокчейну порожню відповідь, що інтерпретовано як відсутність дозволу, та повернув 403 Forbidden.
- **Спроба підміни хешу в off-chain сховищі.** Безпосередньо в PostgreSQL виконано UPDATE-запит, що змінив зміст одного з

документів. При наступному читанні бекенд лікарні обчислив SHA-256 від нового змісту, порівняв з on-chain хешем, виявив розбіжність та повернув 409 Conflict з відповідним повідомленням про порушення цілісності.

- **Запит з простроченим дозволом.** Налаштовано дозвіл з expires_at через 60 секунд, після спливу часу лікар спробував зчитати документ. Логіка модуля x/access правильно інтерпретувала прострочений дозвіл як відсутній; запит повернув 403 Forbidden.
- **Cross-chain replay.** Сформовано підпис у тестовій мережі з chain_id = "medchain-test-1" та надіслано на бекенд, налаштований на роботу з chain_id = "medchain-mainnet-1". Перевірка chain_id коректно відхилила запит з 401 Unauthorized.

Усі описані сценарії атак були коректно нейтралізовані відповідними механізмами захисту. Це підтверджує практичну дієздатність обраної архітектури: комбінація on-chain контролю прав, ADR-036-автентифікації та content-addressable storage забезпечує цілісність та керованість доступу до медичних даних на рівні криптографічних гарантій, без необхідності довіри до окремих компонентів системи (адміністраторів лікарень, мережних посередників).

Окремо проведено перформанс-тестування. У сценарії 50 одночасних запитів на читання документів середній час відповіді API склав 320 мс, 95-й перцентиль - 480 мс, 99-й перцентиль - 540 мс. Затримка анкорування хешу (від моменту формування транзакції до її включення в блок) у середньому склала 2,8 с при налаштуванні CometBFT з інтервалом блоків 1 с. Пропускна здатність публікації хешів - близько 80 транзакцій на секунду в одновалідаторній мережі, що з запасом перекриває реалістичні навантаження медичного консорціуму (у середньому потужний регіональний госпіталь генерує до кількох сотень нових документів на день).

Тестування розробленої системи можна вважати успішним - усі заплановані функціональні сценарії відпрацьовано коректно, механізми захисту витримали цілеспрямовані атаки, показники продуктивності відповідають цільовим. Для подальшого розгортання у виробничих умовах необхідно провести зовнішній аудит коду блокчейн-модулів, налаштувати моніторинг (Prometheus + Grafana для метрик ноди, ELK для журналів бекенду) та сформувати формальні політики управління ключами адміністраторів та підтримуваним списком лікарень.

2.5 Перспективи подальшого розвитку

Розроблений прототип демонструє життєздатність обраного підходу, але не вичерпує всіх можливостей застосування блокчейн-технологій у сфері охорони здоров'я. Розглянемо кілька напрямів подальшого розвитку, які природно вписуються в сформовану архітектуру:

- **Інтеграція з HL7 FHIR**

HL7 FHIR (Fast Healthcare Interoperability Resources)[9] - сучасний стандарт обміну медичною інформацією, прийнятий у переважній більшості країн з розвиненою цифровою медициною. Стандарт визначає схеми Resource-ів (Patient, Practitioner, Observation, Encounter, Condition тощо) та формат їх обміну через REST API. Інтеграція розробленого бекенд-сервісу лікарні з FHIR-серверами дозволить безшовно вписати запропоновану систему у наявну медичну IT-інфраструктуру: документи, що зберігаються off-chain, можна структурувати за схемами FHIR, а REST API лікарні - зробити FHIR-сумісним. Хеш документа в такому випадку обчислюється від канонічного JSON-подання FHIR-Resource.

- **Міжмережевий обмін через IBC**

Кожен медичний консорціум - група лікарень, що домовилися спільно експлуатувати мережу, є самодостатнім блокчейном із власним набором валідаторів. Однак у реальному світі пацієнт може лікуватися як у державних поліклініках, так і в приватних клініках, як в Україні, так і за кордоном. Міжмережевий обмін через протокол IBC (Inter-Blockchain Communication) дозволить різним медичним консорціумам обмінюватися даними про спільних пацієнтів без необхідності створювати єдину гігантську мережу. Зокрема, можливі такі сценарії:

- медичний запис, створений у мережі регіонального консорціуму А, може бути перевірений на цілісність у мережі консорціуму Б через IBC-канал;

- дозвіл, виданий пацієнтом у мережі А, може транслюватися в мережу Б для отримання документів, що зберігаються в офф-чейн сховищах закладів-учасників мережі Б;
- страхова компанія, що працює в окремій мережі, може отримувати IBC-повідомлення про настання страхових випадків (наприклад, виписка пацієнта зі стаціонару).

Реалізація IBC-інтерфейсу в модулях x/access та x/document потребує додаткової роботи з обробки cross-chain транзакцій, але архітектурно вписується в існуючу декомпозицію без необхідності переробляти базову логіку.

- **Підтримка ролі страхових компаній**

Поточна модель дозволів є двосторонньою (пацієнт - лікар), але реальна охорона здоров'я часто включає третю сторону - страхову компанію, яка має право бачити агреговану інформацію про надані послуги для розрахунків і виявлення зловживань, але не повинна мати доступу до повного клінічного контексту. Розширення модуля x/access на роль Insurer із власним набором прав (наприклад, READ_BILLING без READ_CLINICAL) дозволить вписати страхові компанії в систему природним чином, без створення окремих обхідних каналів.

- **Гомоморфне шифрування для агрегованої статистики**

Однією з фундаментальних задач у медичній інформатиці є проведення статистичних досліджень на великих датасетах без розкриття індивідуальних записів пацієнтів. Класичні підходи (анонімізація, k-anonymity) мають обмеження - деанонімізація через зовнішні дані залишається можливою. Гомоморфне шифрування дозволяє виконувати обчислення (підрахунок середніх, медіан, кореляцій) безпосередньо над зашифрованими даними, з результатом, що розшифровується лише уповноваженою стороною. Інтеграція гомоморфного шифрування у бекенд-сервіс лікарні дозволила б надавати дослідницьким установам

зашифровані датасети для агрегованого аналізу, з on-chain журналом усіх запитів дослідників для прозорості та аудиту.

- **Post-quantum криптографія**

Дискусія про загрозу квантових комп'ютерів для класичних криптографічних схем (RSA, ECDSA, secp256k1) є актуальною темою останніх років. Хоча практичні квантові атаки на secp256k1 поки що не реалізовані, медичні дані за своєю природою мають довгий горизонт зберігання - записи дитинства можуть бути актуальними через десятки років. Перехід Cosmos SDK на post-quantum-стійкі схеми підпису (CRYSTALS-Dilithium, SPHINCS+) у перспективі 5–10 років зробить розроблену систему стійкою до загроз майбутнього без необхідності переписувати бізнес-логіку - заміна торкнеться лише крипто-провайдера.

- **Self-sovereign identity та інтеграція з державним реєстром**

В Україні активно розвиваються національні цифрові ініціативи (зокрема, «Дія»). Природним кроком розвитку є інтеграція Cosmos-облікових записів пацієнтів з державним реєстром цифрових ідентичностей. Це дозволило б використовувати державну верифікацію особи для первинного підтвердження пацієнта в медичній блокчейн-мережі, уникаючи створення окремого довіреного посередника для KYC.

- **Мобільний клієнтський застосунок**

Поточний прототип реалізовано без графічного інтерфейсу - клієнтська взаємодія проводиться через CLI та REST API. Природним продовженням є розробка мобільного застосунку, що поєднував би функції криптогаманця (зберігання приватного ключа пацієнта в Secure Enclave / Android Keystore), перегляду активних дозволів та керування ними (видача, відкликання), а також зручний UI для перегляду власної медичної історії. Бекенд лікарні вже надає всі необхідні API для такого клієнта, а його розробка потребує переважно роботи з UX, не з архітектурою.

Загалом, обрана архітектура на базі Cosmos SDK залишає широкий простір для еволюції без необхідності переробляти базову модель. Завдяки модульній структурі, кожне з перерахованих розширень реалізується додаванням нового кастомного модуля або інтеграційного шару, без впливу на вже працюючу частину системи. Це робить запропонований підхід не лише життєздатним прототипом, але й перспективною основою для побудови повноцінної національної екосистеми управління медичними даними.

Висновки

В ході магістерської дипломної роботи були досліджені проблеми сучасних централізованих систем зберігання електронних медичних записів, проаналізовані можливості блокчейн-технологій для їх вирішення, обґрунтовано вибір Cosmos SDK як платформи реалізації та спроєктовано і розроблено працюючий прототип децентралізованої системи з пацієнтоорієнтованим керуванням доступом до медичних даних.

Виконано глибокий аналіз предметної області. Виявлено три фундаментальні проблеми класичних EHR-систем - концентрація прав власності на дані у закладу-операторі замість самого пацієнта, відсутність програмних гарантій від тихих модифікацій записів адміністраторами та залежність міжзакладного обміну від довіреного посередника. Розглянуто існуючі блокчейн-рішення (MedRec, Medicalchain, MyHealthMyData, OmniPHR та інші), визначено їхні архітектурні обмеження - переважна більшість використовує універсальні смарт-контрактні платформи з відповідними накладними витратами, бінарні моделі дозволів без гранулярності за типом операції та областю дії, відсутність нативної підтримки міжмережевої взаємодії.

Обґрунтовано вибір Cosmos SDK як платформи реалізації. На основі порівняльного аналізу з Ethereum та Hyperledger Fabric за критеріями відсутності накладних витрат віртуальної машини, миттєвої фінальності блоків через CometBFT, нативної підтримки permissioned-режиму та гнучкості моделі стану через прямий KVStore-доступ продемонстровано переваги прикладно-специфічного блокчейну для задач медичної інформатики. Окремо відзначено перспективу майбутньої інтеграції в єдину екосистему медичних мереж через стандартизований протокол IBC, що принципово відрізняє запропонований підхід від ізольованих рішень на конкурентних платформах.

Спроєктовано двошарову гібридну архітектуру системи з чітким розділенням on-chain і off-chain відповідальностей. На блокчейні зберігаються виключно реєстри суб'єктів (лікарень, лікарів, пацієнтів), пацієнт-керовані дозволи доступу та криптографічні хеші медичних документів; самі документи знаходяться в off-chain сховищах відповідних медичних закладів. Така архітектура одночасно зберігає переваги блокчейн-технології (гарантована цілісність, прозорість, децентралізований контроль доступу) та задовольняє вимоги до мінімізації обробки персональних даних, закріплені в GDPR та Законі України «Про захист персональних даних», включно з можливістю реалізації права на забуття через off-chain видалення з лишковими знеособленими хешами on-chain.

Розроблено та реалізовано п'ять кастомних модулів Cosmos SDK мовою Go: x/hospital, x/doctor, x/patient, x/access та x/document. Ключовим внеском є модуль x/access з пацієнтоорієнтованою моделлю керування доступом, у якій дозвіл представлено кортежем (patient, doctor, scope, rights, expires_at) з гранулярністю на рівні комбінацій прав CREATE/READ/UPDATE та двома рівнями області дії - окремий документ або вся медична картка пацієнта в межах конкретного закладу. Така модель забезпечує гнучкість видачі прав, недосяжну в існуючих рішеннях на базі Ethereum смарт-контрактів. Модуль x/document забезпечує іммутабельне версіонування медичних документів - кожна модифікація створює нову версію без перезапису попередніх, що зберігає повну історію змін доступною для аудиту. Усі модулі описані через Protocol Buffers з автоматичною генерацією gRPC-серверного коду, що забезпечує відтворюваність і мовну незалежність клієнтських інтеграцій.

Розроблено бекенд-сервіс лікарні мовою Go, що виступає довіреним посередником між блокчейном та off-chain сховищем медичних документів. Сервіс реалізує REST/gRPC API для створення, читання та

оновлення документів, верифікує підписи запитів лікарів за стандартом ADR-036 (Cosmos Signed Messages), звіряє права доступу зі станом блокчейну через gRPC-запити до повної ноди мережі та забезпечує суворий порядок операцій - спочатку синхронне анкорування хешу документа on-chain і тільки після підтвердження транзакції збереження вмісту в off-chain сховище (PostgreSQL для структурованих клінічних даних та MinIO для бінарних артефактів). Обчислення SHA-256 від збереженого вмісту з порівнянням з on-chain хешем при кожному читанні забезпечує негайне виявлення будь-якої несанкціонованої модифікації даних у off-chain сховищі - фундаментальну гарантію цілісності, недосяжну в класичних централізованих EHR-системах.

Спроєктовано та реалізовано протокол криптографічної автентифікації запитів лікарів на основі стандарту ADR-036. Підписаний payload містить поля chain_id (захист від cross-chain replay), sequence (захист від replay-атак через монотонний лічильник), operation (захист від маніпуляції типом запиту), ідентифікацію цільового ресурсу та issued_at (захист від staleness-атак). Використання єдиного криптографічного контексту - того самого ключа secp256k1, яким підписуються блокчейн-транзакції - усуває необхідність у паралельній системі автентифікації (паролі, JWT-токени) та забезпечує криптографічну несумісну прив'язку дій лікаря до його on-chain ідентичності.

Підтверджено життєздатність запропонованого підходу через комплексне тестування. Проведено функціональне тестування ключових сценаріїв роботи системи - реєстрації суб'єктів, видачі та відкликання дозволів, анкорування документів, передачі пацієнта між лікарями, перегляду історії змін документів. Проведено цілеспрямоване тестування механізмів захисту - спроби replay-атак (нейтралізовано через sequence-контроль), запитів від лікарів з відкликаними дозволами (нейтралізовано через on-chain перевірку), підміни даних в off-chain

сховищі (нейтралізовано через звірку SHA-256 хешів), використання прострочених дозволів (нейтралізовано через перевірку expires_at), cross-chain replay (нейтралізовано через перевірку chain_id). Усі описані сценарії атак коректно нейтралізовані відповідними механізмами захисту, що підтверджує практичну дієздатність обраної архітектури.

Експериментальна оцінка продуктивності продемонструвала відповідність системи цільовим показникам - середній час відповіді API при 50 одночасних запитах склав 320 мс (95-й перцентиль — 480 мс), затримка анкорування хешу — близько 2,8 секунди при налаштуванні CometBFT з інтервалом блоків 1 секунда, пропускна здатність публікації хешів — близько 80 транзакцій на секунду в одновалідаторній мережі, що з запасом перекриває реалістичні навантаження медичного консорціуму.

Реалізовано комплексну CI/CD-інфраструктуру через GitHub Actions з автоматичним запуском юніт-тестів, інтеграційних тестів, статичного аналізу (golangci-lint, gosec) та go test -race для виявлення race-умов. Підготовлено Docker Compose конфігурацію, що піднімає повне середовище мережі з кількома валідаторами, бекенд-сервісами лікарень, базами даних PostgreSQL та об'єктними сховищами MinIO однією командою, що забезпечує повну відтворюваність експериментів та спрощує демонстрацію роботи системи.

В сучасному світі, коли цифрова трансформація охорони здоров'я набирає темпів, а вимоги до захисту персональних медичних даних посилюються нормативно-правовими актами на рівні ЄС (GDPR), США (HIPAA) та України, технологія блокчейну стає дедалі важливішим інструментом побудови систем нового покоління. Інтеграція цієї технології у медичну інформатику через прикладно-специфічні блокчейни на базі Cosmos SDK надає принципово нові можливості - від гарантованої цілісності та прозорого аудиту до пацієнтоорієнтованого керування доступом і безшовного міжзакладного обміну даними. Саме завдяки

модульній архітектурі Cosmos SDK та можливостям майбутньої міжмережевої взаємодії через IBC, обраний підхід є перспективнішим, ніж рішення на базі універсальних смарт-контрактних платформ.

У роботі представлено перспективи подальшого розвитку запропонованої системи, що природно вписуються у сформовану архітектуру без необхідності переробляти базову модель: інтеграція зі стандартом HL7 FHIR для безшовного включення в наявну медичну IT-інфраструктуру, міжмережевий обмін через IBC для об'єднання різних медичних консорціумів у єдину екосистему, підтримка ролі страхових компаній з обмеженою областю доступу, гомоморфне шифрування для конфіденційної агрегованої статистики, перехід на post-quantum-стійкі схеми підпису у міру дозрівання відповідних стандартів NIST, інтеграція з державними цифровими ідентичностями (зокрема, з системою «Дія») та розробка мобільного клієнтського застосунку для пацієнтів. Кожен з цих напрямів реалізується додаванням нового кастомного модуля або інтеграційного шару без впливу на вже працюючу частину системи.

Дана робота є першим кроком до повноцінної інтеграції блокчейн-технологій у національну екосистему управління медичними даними. Вона може бути використана як основа для подальшого розвитку як у теоретичному плані - формальна верифікація властивостей моделі контролю доступу, дослідження приватності метаданих, розширення моделі загроз - так і в практичному, через пілотне розгортання в умовах реального медичного консорціуму, інтеграцію з існуючими медичними інформаційними системами та масштабування на національний рівень. Отримані результати мають як теоретичну вагу для подальших наукових досліджень у сфері blockchain-based EHR, так і практичне значення для побудови реальних систем управління медичними записами нового покоління.

Список використаних джерел

1. Azaria A., Ekblaw A., Vieira T., Lippman A. MedRec: Using Blockchain for Medical Data Access and Permission Management // Proceedings of the 2nd International Conference on Open and Big Data (OBD). — IEEE, 2016. — P. 25–30. — URL: https://ieeexplore.ieee.org/document/7573685 (дата звернення: 15.02.2026).
2. Medicalchain Whitepaper v2.1: веб-сайт. URL: https://medicalchain.com/Medicalchain-Whitepaper-EN.pdf(дата звернення: 18.02.2026).
3. MyHealthMyData (MHMD) Project. Horizon 2020 Programme: веб-сайт. URL: https://cordis.europa.eu/project/id/732907 (дата звернення: 20.02.2026).
4. Roehrs A., da Costa C. A., da Rosa Righi R. OmniPHR: A distributed architecture model to integrate personal health records // Journal of Biomedical Informatics. — 2017. — Vol. 71. — P. 70–81. — URL: https://www.sciencedirect.com/science/article/pii/S1532046417301089 (дата звернення: 22.02.2026).
5. Cosmos SDK Documentation. High-level overview: веб-сайт. URL: https://docs.cosmos.network/main/learn/intro/overview (дата звернення: 05.02.2026).
6. CometBFT Documentation. Introduction: веб-сайт. URL: https://docs.cometbft.com/v0.38/introduction/ (дата звернення: 08.02.2026).
7. IBC Protocol Documentation. Overview: веб-сайт. URL: https://ibc.cosmos.network/main/ibc/overview (дата звернення: 12.02.2026).

8. ADR-036: Arbitrary Message Signature Specification // Cosmos SDK Architecture Decision Records: веб-сайт. URL: https://github.com/cosmos/cosmos-sdk/blob/main/docs/architecture/adr-036-arbitrary-signature.md (дата звернення: 28.02.2026).
9. HL7 FHIR Release 5 Specification // Health Level Seven International: веб-сайт. URL: https://hl7.org/fhir/R5/(дата звернення: 25.02.2026).
10. Kwon J., Buchman E. Cosmos: A Network of Distributed Ledgers. — 2016. — URL: https://v1.cosmos.network/resources/whitepaper (дата звернення: 03.03.2026).
11. Hyperledger Fabric Documentation: веб-сайт. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.5/(дата звернення: 14.02.2026).
12. Androulaki E., Barger A., Bortnikov V. et al. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains // Proceedings of the Thirteenth EuroSys Conference. — ACM, 2018. — P. 1–15.
13. Buchman E., Kwon J., Milosevic Z. The latest gossip on BFT consensus. — 2018. — URL: https://arxiv.org/abs/1807.04938 (дата звернення: 09.02.2026).
14. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation) // Official Journal of the European Union. — 2016. — URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj (дата звернення: 20.01.2026).
15. Основи законодавства України про охорону здоров'я : Закон України від 19.11.1992 № 2801-XII (зі змінами): веб-сайт. URL: https://zakon.rada.gov.ua/laws/show/2801-12 (дата звернення: 21.02.2026).

16. U.S. Department of Health and Human Services. Health Insurance Portability and Accountability Act of 1996 (HIPAA): веб-сайт. URL: https://www.hhs.gov/hipaa/index.html (дата звернення: 22.02.2026).
17. ISO/IEC 27799:2016 Health informatics — Information security management in health using ISO/IEC 27002. — Geneva : ISO, 2016. — 42 p.
18. ISO/IEC 27001:2022 Information security, cybersecurity and privacy protection — Information security management systems — Requirements. — Geneva : ISO, 2022. — 19 p.
19. National Institute of Standards and Technology. Post-Quantum Cryptography Standardization: веб-сайт. URL: https://csrc.nist.gov/projects/post-quantum-cryptography (дата звернення: 30.03.2026).

Додаток А

Код ноди cosmos-based мережі зі створеними модулями x/doctor, x/patient, x/document, x/hospital, x/access доступний за посиланням: https://github.com/slandymani/medchain

Код бекенд сервісу лікарні доступний за посиланням: https://github.com/slandymani/hospital-svc

Репозиторій з конфігурацією docker compose для розгортання мережі з блокчейн нодами, сервісами лікарні, базою даних Postgres та сховищем MinIO доступний за посиланням: https://github.com/slandymani/dev-edition-medchain