College Dublin
Computing • IT • Business

Assessment Cover Page

Student Full Name	Carlos Eduardo Borges Torres de Menezes Neto Renan de Castilhos da Silva
Student Number	2023252; 2023211
Module Title	Distributed Digital Transactions
Assessment Title	Decentralized
Lecturer/Supervisor	Muhammed Iqbal
Assessment Due Date	21/11/2025 @ 11:59pm
Date of Submission	24/11/2025

Use of AI Tools

I acknowledge the use of CopilotAI for the purpose of Structuring the texts and correcting the grammar and the use of RemixAI assistant for the purpose of bug fix and deep explanation of the code in order to have a better understanding of the subject.

Declaration

By submitting this assessment, I confirm that I have read the CCT policy on academic misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source.

I declare it to be my own work and that all material from third parties has been appropriately referenced.

I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

Abstract

Contents

Introduction

This assignment has the objective of testing our knowledge in blockchain and smart contracts development by searching about blockchain, encryption techniques and business model along with the development of a decentralized app with smart contract for a specific business model.

For our assignment we've decided to develop a smart contract for a healthcare system and medical records. Where users could create medical records and the records would be assigned to their own address in the blockchain while they would become owners of these records. As owners they have the option of grant or revoke access to other users passing their records Id and the user address in the blockchain. Using the right modifiers, the only users(addresses) allowed to read the medical record data would be the owner and whoever has access to it.

For the inputs the user has to use the cid of the data it wants to create by using IPFS encryption along with the SHA256 hash for the same file. When another user has access to the file through the grant access function from the contract, it can have access to the cid and hash of the record and then can decrypt and access the file.

We also decided to create a function for other addresses to request access to records using the id of the file and when doing it, the owner would be notified.

The assignment also requested us to discuss about the benefits and challenges of applying blockchain to this specific use case and identify the most suitable cryptographic method for securing information in this scenario explaining how it enhances security in the blockchain

Cryptographic method for healthcare and medical records

There are plenty of benefits in applying blockchain to medical records, some of them include:

- Tamper-evident audit trail

    This include immutable event logs and transactions provide a verifiable timeline of record creation, access requests, grants and revocations that supports compliance and audit.

- Patient-centric access control
    - This guarantees that the patient(creator/owner of the record) will keep his on-chain authority for access decisions, enabling stronger consent than other systems.
- Interoperability enabler
    - Common on-chain metadata surface. Allows multiple providers and apps to operate independently.
- Reduced single-point-of-failure
    - Provides decentralized ledger validation reducing risks from compromised central database and helps detect unauthorized changes.
- Off-chain data minimization
    - Keeps sensitive information and data off-chain by storing only metadata on-chain while retaining provable integrity and access control references.

Most suitable cryptographic method for Healthcare and medical records

Based on research and previous cases, the most recommended approach for this business case would be a hybrid encryption combining symmetric(AES-GCM or ChaCha20-Poly1305) for content and asymmetric key agreement and encapsulation for secure key distribution along with on-chain commitments using keccak256.

This approach is more recommended given that symmetric algorithms such as AES-GCM and ChaCha20-Poly are fast and more recommended for off-chain encrypting large files like images and records. They also provide confidentiality and built-in integrity/authentication to detect tempering. The Asymmetric key agreement such ECDH/ECIES lets a patient encrypt the symmetric content key for one or many recipients without the need of re-encrypting the whole file each time.

On-chainwise is recommended to store only a content commitment and integrity hash, usually keccak256 or SHA-256, and access flags. plaintext or symmetric keys are never recommended to be stored. The contentHash ties an on-chain record to off-chain encrypted content and assures integrity when content is retrieved.

Challenges and limitations

- Privacy leakage from on-chain metadata
    - CIDs or other identifiers could be leaked when stored on-chain so this visibility must be minimized or mitigated.
- Scalability and costs
    - When dealing with blockchain, the per-record transactions, especially for large patient populations, are very expensive. The higher the volume, higher the gas and consequently, higher the cost.
- Usability and key management
    - Patients and clinicians/doctors must know how to manage their keys securely. Lost keys can permanently block access to records unless there's a recovery mechanism implemented.
- Off-chain dependence and trust
    - Encrypted off-chain content can't be protected by blockchain alone. Measures like a robust off-chain storage, access delivery systems and key rotation are required in order to prevent data leak or access breach.
- Emerging threats and future-proofing
    - Schemes like ECDSA and RSA face future quantum threats requiring long-term health data post-quantum planning.

How the IFPS method enhances security in the blockchain

Evaluating citation usage

I'm thinking that I should include citations after relevant paragraphs, but I want to ensure they're numbered sequentially, and only when using info from external sources. Maybe I can try citing things like Kaleido, Webopedia, Robots.net, and

Nadcab, making sure I don’t overcomplicate it. I definitely want to keep each paragraph concise, so I’ll include citations only where needed to support the info. I'll make sure each paragraph that references external facts ends with a citation.

Overview

IPFS (InterPlanetary File System) is a content-addressed, peer-to-peer storage network that stores and retrieves files by cryptographic CID (content identifier) rather than by location. Using IPFS alongside a blockchain lets the chain keep small, auditable references (CIDs and integrity hashes) while large, sensitive medical files remain off-chain, improving scalability and cost-efficiency.

Key benefits of using IPFS with your MedicalRecords contract

- Content integrity and tamper-evidence: IPFS CIDs are derived from cryptographic hashes of file content, so any off-chain alteration changes the CID; the on-chain contentHash or CID therefore binds on-chain metadata to the exact off-chain ciphertext and makes tampering detectable.
- Decentralized availability and redundancy: files are served by many nodes in the network (not a single server), improving resilience to single-point failures and censorship compared with centralized storage.
- Efficient on-chain footprint: storing only CIDs and hashes on-chain keeps gas costs low while preserving a verifiable link to the full record stored off-chain.
- Native deduplication and content addressing: identical files map to the same CID, saving storage and simplifying integrity checks across providers.

How IPFS enhances security in the blockchain architecture

- Strong cryptographic binding: because IPFS uses content-based addressing (hash-derived CIDs), the on-chain record can reference an immutable commitment to the exact bytes stored off-chain. Verifying the retrieved file

against the CID/contentHash yields immediate detection of tampering or substitution.

- Separation of concerns (confidentiality vs. auditability): encryption preserves confidentiality off-chain (so public CIDs never reveal plaintext), while IPFS + blockchain provide immutable auditability of who created or authorized access to a particular encrypted object.
- Defenses against availability attacks: decentralized pinning and multiple peers increase the chance the ciphertext remains retrievable even when some nodes or providers are down, supporting reliable off-chain retrieval for authorized parties.
- Lightweight on-chain proofs: small cryptographic commitments stored on-chain (CID, keccak256/SHA-256) provide cheap, verifiable proofs that can be referenced in consent workflows and regulatory audits.

Practical complements that make IPFS secure in production

- Always encrypt medical content before pinning. Treat IPFS as encrypted storage plus content-addressed integrity rather than a confidentiality mechanism; encryption keys must be managed off-chain and distributed only to authorized grantees (wrapped per the hybrid scheme discussed earlier).
- Pinning strategies and service-level guarantees. Use managed pinning services, decentralized pinning (multiple providers), or institutional nodes to ensure data persists; otherwise garbage collection or node churn can make data unavailable despite a stored CID.
- Access-release gating. Combine on-chain canAccess checks and events with an off-chain gateway or access service that only releases the encrypted symmetric key (or decrypts for the grantee) after verifying the on-chain permission state.

- Use commitments and short opaque references when privacy sensitivity of CIDs is a concern (store salted commitments or hashed pointers on-chain rather than raw CIDs) to reduce correlation risk.

Limitations, risks and mitigations

- Metadata leakage: even encrypted CIDs and announcement events can be correlated to reveal usage patterns or link identities; mitigate via pseudonymization, minimal on-chain metadata, and careful off-chain indexing.
- Persistence is not automatic: IPFS does not guarantee indefinite storage unless content is pinned; plan and fund pinning across trustworthy nodes or services to avoid data loss.
- Discovery and availability trade-offs with gateways: public gateways simplify retrieval but add trust and privacy surface; prefer application-controlled gateways or authenticated peers for sensitive records.
- Regulatory constraints: immutable on-chain references may complicate erasure/“right to be forgotten”; address this by keeping only minimal commitments on-chain and managing deletions and retention off-chain in compliance workflows.

Design principles used on the project

Principle 1: Design Before Code

Provide a simple on-chain access-controlled medical-records registry where:

- Patients (owners) register encrypted or IPFS-hosted records with content hash and timestamp.
- Patients control granular read access per record by granting and revoking addresses.
- Access requests are recorded as events so off-chain workflows (consent UI, audit) can surface requests and approvals.
- The contract stores metadata only (CID, contentHash, owner, createdAt); actual medical data remains off-chain and encrypted.

Participants (who interacts)

- Patient (Record owner): creates records; grants and revokes access; always has implicit access.
- Requester (e.g., clinician, researcher): requests access; reads metadata only if granted off-chain.
- Auditor / Anyone: reads public metadata and checks accessGranted flags; listens to events for audit and off-chain flows.

Testing addresses (Remix default example)

- ACCOUNT 1: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
- ACCOUNT 2: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

- ACCOUNT 3: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db

Scope / Non-scope

In scope

- Patient-only record creation (createRecord).
- Off-chain request lifecycle signalled by events (AccessRequested).
- Patient-managed access control (grantAccess, revokeAccess).
- View function for access check (canAccess) and public read of record metadata.
- Minimal on-chain storage: metadata only (no plaintext medical data).

Out of scope

- Storing plaintext medical data on-chain.
- Fine-grained attribute-level permissions (read vs annotate vs share) — only simple boolean access per address.
- Role-based or multi-owner governance, emergency access overrides, or consent revocation delegation.
- Complex consent workflows (time-bound access, purpose-limited access) unless extended off-chain or with additional fields.
- Payments, billing, or identity verification (KYC) on-chain.
- Audit log immutability for off-chain content — events provide trace but off-chain correlation required.

Data model (high level)

- records : mapping(uint256 => Record) where Record { owner: address; cid: string; contentHash: bytes32; createdAt: uint256 }
- accessGranted : mapping(uint256 => mapping(address => bool)) boolean granting read access to a specific address per record.
- nextRecordId : uint256 auto-incrementing record identifier.

Key invariant: records[id].owner != address(0) ⇨ record exists.

Authoritative state machine (access lifecycle)

Because control is owner-centric and access is boolean toggles, model the high-level access lifecycle per record and per address.

States (per record per address)

- NoAccess: accessGranted[id][addr] == false and addr != owner.
- Requested: off-chain notion represented by AccessRequested event; on-chain state unchanged.
- Granted: accessGranted[id][addr] == true (owner granted).
- Revoked: accessGranted[id][addr] == false after a revoke (may be same as NoAccess but semantically a revocation event exists).

Transitions (triggering events/txs)

- requestAccess (external call): NoAccess or Revoked -> emits AccessRequested (off-chain workflows pick up)
- grantAccess (owner only): NoAccess/Requested/Revoked -> Granted (emit AccessGranted)
- revokeAccess (owner only): Granted -> Revoked (emit AccessRevoked)
- owner implicit access: owner is always considered in Granted state for canAccess checks

Notes

- Events serve as the canonical audit trail for requests and decisions; on-chain mapping records current access boolean only.
- No temporal constraints or purpose-scoped transitions — adding those would require extra fields (expiry, purpose hash).

Principle 2: Users, Use Cases, Rules and Transactions

Actors and concise use cases

- Patient (owner)
    - CreateRecord(cid, contentHash) -> store metadata and emit RecordCreated
    - GrantAccess(recordId, grantee) -> set accessGranted true and emit AccessGranted
    - RevokeAccess(recordId, grantee) -> set accessGranted false and emit AccessRevoked
    - View own records and canAccess checks
- Requester
    - RequestAccess(recordId) -> emit AccessRequested for off-chain approval workflow
    - Query canAccess(recordId, requester) -> view whether access exists
- Auditor / Anyone
    - Read public record metadata (owner, cid, contentHash, createdAt)
    - Listen to events for off-chain logging and compliance

Data (table-style summary)

- nextRecordId : uint256 — ID counter for records
- records[id] : Record — metadata about the record
- Record.owner : address — immutable patient address for that record

- Record.cid : string — pointer to encrypted off-chain data (IPFS/URI)
- Record.contentHash : bytes32 — integrity hash of off-chain content
- Record.createdAt : uint256 — timestamp of creation
- accessGranted[id][address] : bool — current granted access flag

Access control rules

- onlyOwner(id) modifier: only records[id].owner can grant or revoke access and must be enforced on grantAccess/revokeAccess.
- Existence check: functions that act on a record must verify record exists (records[id].owner != address(0)).
- canAccess returns true if user == owner or accessGranted flag is true.
- Events used for off-chain workflows: RecordCreated, AccessRequested, AccessGranted, AccessRevoked.

Safety rules and best practices

- Minimal on-chain data: store only metadata; actual medical content remains off-chain and encrypted by patient.
- Validate existence before state changes to avoid accidental creation or acting on non-existent ids.
- Use indexed event fields for efficient off-chain filtering (owner, recordId, requester/grantee).
- Prefer pull-based data retrieval for sensitive content: on-chain access flag only authorizes off-chain mechanisms to release decrypted content to the grantee.
- Consider meta-consent patterns off-chain (signed consent objects) to reduce on-chain operations and gas costs.
- Protect against griefing: consider rate-limiting or economic deterrents for spammy requestAccess calls if necessary (out of scope in current simple design).

Transactions vs Views

- createRecord(cid, contentHash) external returns (uint256) — Transaction; mints metadata and emits RecordCreated.
- requestAccess(uint256 id) external — Transaction; emits AccessRequested for off-chain workflows.
- grantAccess(uint256 id, address grantee) external onlyOwner(id) — Transaction; sets access flag true and emits AccessGranted.
- revokeAccess(uint256 id, address grantee) external onlyOwner(id) — Transaction; sets access flag false and emits AccessRevoked.
- canAccess(uint256 id, address user) public view returns (bool) — View; returns current permission boolean.
- records(uint256) public view — auto getter returns Record metadata.

Principle 3: Contract Diagram, Events and Operational Notes

Contract: MedicalRecords Attributes

- nextRecordId : uint256
- records : mapping(uint256 => Record)
- accessGranted : mapping(uint256 => mapping(address => bool)) Record struct
- owner : address
- cid : string
- contentHash : bytes32
- createdAt : uint256

Events

- RecordCreated(recordId, owner, cid, contentHash) — index recordId, owner
- AccessRequested(recordId, requester) — index recordId, requester
- AccessGranted(recordId, grantee, by) — index recordId, grantee, by

- AccessRevoked(recordId, grantee, by) — index recordId, grantee, by

Principle 4: Access Rules

- onlyOwner modifier enforces patient control over grant/revoke operations.
- canAccess is the canonical on-chain check for permission; off-chain services must rely on this boolean plus event history.

Operational notes and extensions

- Off-chain integration: An access-granted workflow should require an off-chain service to verify canAccess before releasing decrypted content to the grantee. The contract does not and must not manage decryption keys.
- Privacy: cid is visible on-chain; use an opaque identifier or double-layer encryption so on-chain CID exposure doesn't leak meaningful info. Consider storing only a short commitment on-chain and keeping full CID off-chain if metadata sensitivity is a concern.
- Auditability: Events provide a tamper-evident timeline of requests and grants; store event logs in an off-chain index for user-facing audit and consent records.
- Gas and UX: Creating many small records can be expensive; consider batching or merkle-root approaches for high-volume scenarios.
- Extensions (future): time-limited grants (expiry timestamp), purpose-scoped grants (hash of allowed purpose), delegated consent, emergency access with multi-sig, or revocation receipts.

Security checklist (pre-deployment)

- Verify onlyOwner uses correct record existence check.
- Ensure no function unintentionally writes to records mapping (only createRecord should set records[id]).
- Test canAccess logic for owner implicit access.
- Add input size/length checks for cid to avoid gas or storage anomalies.
- Consider events indexed fields for efficient off-chain filtering.

References

Bak, Ozlem, et al. “Exploring Blockchain Implementation Challenges in the Context of Healthcare Supply Chain (HCSC).” Ebsco.com, International Journal of Production Research, Jan. 2025, research.ebsco.com/c/jzntuu/viewer/html/5ysjij5smf. Accessed 9 Nov. 2025.

“Blockchain in Healthcare: Benefits, Use Cases and Challenges.” TMA Solutions, 2022, www.tmasolutions.com/insights/blockchain-in-healthcare.

Dou, Tengyue, et al. “A Secure Medical Data Framework Integrating Blockchain and Edge Computing: An Attribute-Based Signcryption Approach.” Sensors, vol. 25, no. 9, 30 Apr. 2025, pp. 2859–2859, www.mdpi.com/1424-8220/25/9/2859, https://doi.org/10.3390/s25092859. Accessed 21 Nov. 2025.

Jabri, Abdou-Essamad, et al. “Leveraging Blockchain and Proxy Re-Encryption to Secure Medical IoT Records.” ArXiv.org, 2025, arxiv.org/abs/2509.08402. Accessed 20 Nov. 2025.

Katru Rama Rao, and Satuluri Naganjaneyulu. “Designing a Block Chain Based Network for the Secure Exchange of Medical Data in Healthcare Systems.” Applied Artificial Intelligence, vol. 38, no. 1, 11 Mar. 2024, https://doi.org/10.1080/08839514.2024.2318164. Accessed 16 Nov. 2025.

Khouzima_hamza. “Implementation of IPFS in Healthcare Projects: A Comprehensive Overview.” Medium, 14 Mar. 2024, medium.com/@khouzimahamza20/implementation-of-ipfs-in-healthcare-projects-a-comprehensive-overview-f0afbdf7e82c. Accessed 18 Nov. 2025.

Malik, Pankaj, et al. “Integrating Blockchain and 5G Technologies for Enhanced Edge Computing: Opportunities, Challenges, and Solutions.” International Journal of Recent Advances in Multidisciplinary Topics, vol. 5, no. 2, 2024, pp. 35–43, journals.ijramt.com/index.php/ijramt/article/view/2855, https://doi.org/10.5281/zenodo.10702502. Accessed 9 Nov. 2025.

Megha, Jain, and Pandey Dhiraj. “HRCM: An Approach Using Blockchain Technology in Healthcare-Record Chain Management.” Ebsco.com, International Journal of Performability Engineering, Jan. 2025, research.ebsco.com/c/jzntuu/viewer/pdf/4adi773hvr. Accessed 20 Nov. 2025.

Mishra, Debani Prasad, et al. “Efficient Blockchain Based Solution for Secure Medical Record Management.” International Journal of Informatics and Communication Technology (IJ-ICT), vol. 14, no. 1, 22 Dec. 2024, pp. 59–59, https://doi.org/10.11591/ijict.v14i1.pp59-67. Accessed 9 Nov. 2025.

Mohanapriya, Mrs, et al. “Med-Block: Secure Health Record Management System Using Blockchain with Ipfs.” International Journal of Engineering Research & Technology (IJERT), 4 Apr. 2024.

Noor, et al. “Blockchain-Based Healthcare Records Management Framework: Enhancing Security, Privacy, and Interoperability.” Technologies, vol. 12, no. 9, 14 Sept. 2024, pp. 168–168, www.mdpi.com/2227-7080/12/9/168, https://doi.org/10.3390/technologies12090168. Accessed 3 Nov. 2025.

---. “Blockchain-Based Healthcare Records Management Framework: Enhancing Security, Privacy, and Interoperability.” Technologies, vol. 12, no. 9, 14 Sept. 2024, pp. 168–168, www.mdpi.com/2227-7080/12/9/168, https://doi.org/10.3390/technologies12090168. Accessed 20 Nov. 2025.

Pongallu, Darshika Ravindra. “Securing Medical Records Using Blockchain with Cryptography, Encryption, and Zero-Knowledge Rollups.” Ncirl.ie, 2024, norma.ncirl.ie/8049/, https://norma.ncirl.ie/8049/1/darshikaravindrapongallu.pdf. Accessed 21 Nov. 2025.

Singh, Murari Kumar, et al. “A Blockchain-IPFS Framework for Secure, Scalable, and Interoperable Healthcare Data Management.” SN Computer Science, vol. 6, no. 5, 17 Apr. 2025, https://doi.org/10.1007/s42979-025-03936-z. Accessed 9 Nov. 2025.

Tran, Phong, et al. “A Solution for Commercializing, Decentralizing and Storing Electronic Medical Records by Integrating Proxy Re-Encryption, IPFS, and Blockchain.” ArXiv.org, 2024, arxiv.org/abs/2402.05498. Accessed 20 Nov. 2025.

Appendix