National University of Computer and Emerging Sciences
Islamabad Campus

Project Proposal

Secure Digital Health Records System- Patient Centered Encryption

CY4001
Secure Software Design

Submitted by:
CY-C
Eman Akbar 22i-1588
Laiba Waseem 22i-1566
Huda Imran 22i-1713
Shifa Zehra 22i-1727
CY-A
Saad Umar 22i-1679
Date: 31/08/25

National University of Computer and Emerging Sciences
Islamabad Campus

Secure Digital Health Records System – Patient-Centered Encryption

Team Information

Eman Akbar – System Architect I & Frontend Developer (Web Interface)_Team Lead

Laiba Waseem– System Architect II & Documentation Lead & QC

Saad Umar – Backend Developer (Encryption & APIs)

Huda Imran – Backend Developer (Database & Access Control)

Shifa Zehra – Security Analyst (Threat Modeling & Testing)

Problem Statement

Medical records are extremely sensitive but are maintained in centralized systems with very little control by patients, creating the risk of data breaches and unauthorized access. Patients do not have a choice on how their data is encrypted or shared. We propose to develop a system in which patients can securely store, retrieve, and share health records through the use of simple encryption that is under their direct control and will illustrate correct software handling of security considerations in healthcare applications.

Objectives

- Allow simple client-side encryption of health records (AES-256).
- Use role-based access control (RBAC) for patients and doctors.
- Record audit logs of all access attempts.
- Show adherence to HIPAA/GDPR fundamental principles (confidentiality, consent, deletion)
- Demonstrate secure coding principles in the software development lifecycle

Proposed Solution

1) Architecture (MVP)

- **Frontend (React):** Patient/doctor portals; **WebCrypto** for client-side crypto.
- **Backend (Flask/Python): JWT** auth, consent endpoints, basic audit logging, pre-signed upload/download.
- **Storage: PostgreSQL** (users, records, consents, audit_logs; optional **RLS**) + **MinIO** (ciphertexts only).
- **Runtime: Docker Compose** (web, api, db, minio). No cloud needed for MVP.

2) Data & Crypto

- Per file: generate 256-bit key → encrypt with **AES-256-GCM** in browser.
- **PEK** (patient encryption key): derive from passphrase via **PBKDF2** (WebCrypto). Store **salt/params only** server-side.
- **Sharing:** wrap file key with doctor **RSA-OAEP** public key; doctor decrypts locally with private key.
- **Server never sees plaintext** files or keys.

3) Roles & Access

- **Roles:** patient, doctor.
- **Policy:** patients manage own records & consents; doctors **read-only** when **consent active** and a **wrapped key** exists.

4) Consent & Sharing Flow

1. Patient encrypts & uploads → server stores metadata + patient wrap.
2. Patient grants consent (record/folder scope).

3. Client creates **doctor wrap** using doctor public key.
4. Doctor fetches ciphertext + wrap → decrypts locally.
5. Revoke → mark consent inactive & remove doctor wraps.

5) Audit Logging (Patient-Visible)

- Log: **who, action, record, allow/deny, time, IP/UA**.
- Simple dashboard table with filters.

6) Minimal API Surface

POST /auth/login	→ JWT
GET /records	→ own + shared
POST /records	→ save metadata (uri, iv, algo, wraps)
POST /records/presign	→ upload URL
POST /consents	→ grant {doctor_id, record_id|scope}
PATCH /consents/:id	→ revoke
GET /audit	→ patient logs

7) Compliance (Simulated, Clear UX)

Explicit consent, delete record (ciphertext + metadata), no plaintext PHI server-side, full audit transparency.

Methodology (S-SDLC)

1. **Requirements:** minimum compliance features (encryption, consent, logging).
2. **Design:** architecture diagrams + **STRIDE** threat modeling.

3. **Implementation:** secure coding, client-side AES, **JWT** auth.
4. **Testing: SAST (SonarQube)**, **DAST (OWASP ZAP)**, basic pentest on login/APIs.
5. **Deployment:** HTTPS Dockerized setup (Caddy optional).
6. **Maintenance:** simple key rotation & patching.

Tools and Technologies

- **Backend: Flask**, Flask-JWT-Extended, SQLAlchemy/psycopg2.
- **Frontend: React**, WebCrypto API, Axios/Fetch.
- **Security:** AES-256-GCM, PBKDF2 (WebCrypto), server password hashing via **bcrypt/argon2**.
- **Data: PostgreSQL** (+ optional RLS), **MinIO** object storage.
- **Testing: OWASP ZAP**, **SonarQube**.
- **Deploy: Docker Compose** (web, api, db, minio).

Threats & Mitigations (Brief)

- **Auth/Impersonation:** strong passwords, rate-limit/lockout, short-lived JWT, HTTPS.
- **Broken AuthZ/IDOR:** strict server-side checks per record; optional Postgres RLS.
- **Data Exposure:** client-side AES-GCM; ciphertext + minimal metadata only.
- **MITM/Replay:** TLS, HSTS, nonces/timestamps.

- **Key Risks:** PBKDF2 for PEK; optional encrypted PEK backup; rotation re-wraps keys.
- **Consent Bypass:** require active consent **and** doctor wrap; remove wraps on revoke.
- **Audit Integrity:** append-only logs; patient audit UI.

Expected Deliverables

- Working prototype (encrypted storage + consented sharing).
- Design docs (architecture, DFDs), **threat model**, compliance mapping.
- Security testing report (SAST/DAST).
- Final demo & presentation.

Timeline (10 Weeks)

- **W1–2:** Requirements + threat modeling
- **W3–4:** Flask APIs + encryption flow
- **W5–6:** React integration + RBAC/consent
- **W7–8:** Security testing + audit logs
- **W9:** Compliance mapping + docs
- **W10:** Demo & presentation

References

- U.S. Department of Health & Human Services. Summary of the HIPAA Security Rule [Internet]. 2013 [cited 2025 Aug 31]. https://www.hhs.gov/hipaa/for-professionals/security

- European Union. Regulation (EU) 2016/679 (General Data Protection Regulation). Off J Eur Union. 2016 Apr 27.

- OWASP Foundation. OWASP Application Security Verification Standard 4.0.3 [Internet]. 2023. https://owasp.org/ASVS

- OWASP Foundation. OWASP API Security Top 10 – 2023 [Internet]. 2023. https://owasp.org/API-Security

- National Institute of Standards and Technology. Security and Privacy Controls for Information Systems and Organizations. NIST SP 800-53 Rev.5. Gaithersburg (MD): NIST; 2020.

- National Institute of Standards and Technology. Recommendation for Key Management: Part 1 – General. NIST SP 800-57 Pt.1 Rev.5. Gaithersburg (MD): NIST; 2020.