Indonesian Journal of Electrical Engineering and Computer Science
Vol. 41, No. 3, March 2026, pp. 1025∼1039
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v41.i3.pp1025-1039

A smart-contract framework for patient identity management in digital health platforms

Cahyo Prihantoro[1], Dany Candra Febrianto[1], Maie Istighosah[1], Ahmad Uffi Lestrasi Ma'ruf[2], Angger Taufiqur Rohman Winarno[1], Yudha Islami Sulistya[2]
[1]Informatics of Engineering Study Program, Telkom University, Purworkerto Campus, Purwokerto, Indonesia
[2]Software Engineering Study Program, Telkom University, Purworkerto Campus, Purwokerto, Indonesia

Article Info

Article history:

Received Sep 14, 2025
Revised Jan 13, 2026
Accepted Mar 4, 2026

Keywords:

Blockchain
FHIR interoperability
Patient identity management
Self-sovereign identity
Smart contract

ABSTRACT

A smart-contract framework for patient identity management in digital health platforms. A major gap in current digital health ecosystems is the absence of a portable and verifiable patient identity layer across fragmented electronic health record (EHR) systems. The problem addressed is the lack of a portable, verifiable, and patient-centric identity layer across fragmented electronic health record systems, which weakens access accountability and privacy. The proposed solution couples fast healthcare interoperability resources (FHIR) with self-sovereign identity (SSI), storing FHIR payloads off-chain in the InterPlanetary file system (IPFS) and committing only encrypted pointers and policies on Polygon smart contracts. Patient identifiers and content addresses are protected with AES-256-GCM authenticated encryption and elliptic-curve key wrapping (ECIES) for both the healthcare administrator and the patient. A web implementation in Next.js using thirdweb automates wallet creation, keystore handling, encryption, and on-chain commits. In evaluation with 50 synthetic registrations, success reached 100 percent, median end-to-end latency was 5.86 seconds, mean on-chain latency 3.77 seconds, average transaction fee 0.0401 POL/MATIC, encryption time 13.9 milliseconds, and all decryptions validated. The results indicate practical feasibility for portable identity and auditable access, with on-chain latency as the main bottleneck to be reduced through batching, cheaper layers, and broader field trials. However, this study is limited because the evaluation uses only synthetic data and singleprovider testing, without real-world patients or multi-institutional environments. Zero-knowledge proofs (ZKP) are discussed conceptually as future integration and are not implemented or benchmarked in this work.

Corresponding Author:

Cahyo Prihantoro
Informatics of Engineering Study Program, Telkom University, Purwokerto Campus
Street of DI Panjaitan No.128, Purwokerto 53147, Central Java, Indonesia
Email: cahyop@telkomuniversity.ac.id

1. INTRODUCTION

Digital health ecosystems increasingly rely on the seamless exchange of patient data across hospitals, laboratories, insurers, and home devices. However, identity remains a fragile link that determines who may access which data, when, and for what purpose. Centralized identifiers embedded within siloed electronic health records (EHRs) and cloud platforms introduce risks of breaches, opaque provenance, and uneven trust across institutions [1], [2]. Interoperability standards such as fast healthcare interoperability resources (FHIR) facilitate data exchange but, by themselves, do not ensure system trustworthiness or strong identity assurance in heterogeneous internet of medical things (IoMT) environments [3], [4]. As a result, a recurring tension arises between timely care and privacy preservation, especially when medical records are shared across organizations and cross jurisdictional boundaries [5], [6].

However, a fundamental gap persists: current digital health infrastructures lack a portable and verifiable patient identity layer capable of bridging the fragmentation across diverse EHR systems, creating a clear gap that this study aims to address. The problem to be addressed is the absence of a patient identity layer that is truly portable, verifiable, and patient-centric within a fragmented healthcare architecture. Identity remains tied to EHR silos and institutional accounts, making access dependent on central authorities and exposing systems to single points of failure and privilege misuse [2]. Data-exchange standards such as FHIR facilitate interoperability but do not automatically ensure reliability, trust, and identity assurance in heterogeneous, IoMT-rich environments [3]. At the operational level, patient portals and record-sharing mechanisms often lack genuinely fine-grained authorization, robust audit trails for every identity event, and consent evidence that can be easily verified across providers [1], [7]. In the security domain, the Health-IoT ecosystem expands the attack surface and requires a consistent cyber-risk evaluation framework for identity flows, which is not yet established in many implementations [8]. In smart-city and cloud settings, crossservice authentication demands a transparent shared trust model; current centralized approaches do not meet this need [9]. Meanwhile, self-sovereign identity promises direct patient control, but practical deployments that unify portability, privacy, and dynamic consent remain limited [5], [10]-[12]. Together, these barriers drive service delays, duplicative verification, potential data leakage, identity mismatches, and weak access accountability as data move across facilities, jurisdictions, and platforms.

Prior work across digital health highlights recurring needs for secure patient identity, cryptographically enforced access, and standards-based interoperability across institutions [10], [13]-[20]. Although decentralized identity models such as SSI and verifiable credentials offer promising foundations [21]-[23] real deployments remain fragmented and rarely integrate FHIR's structured data model with patientcontrolled decryption. While several studies discuss advanced mechanisms such as dynamic consent or zeroknowledge proofs, these concepts mostly remain theoretical and are not implemented in practice—including in this study [17], [24]-[27]. These gaps motivate the need for a practical, integrated framework that unifies identity, encrypted metadata management, and interoperable clinical data flows.

The proposed approach combines the FHIR standard, self-sovereign identity frameworks, smart contracts on the Polygon network, end-to-end encryption, and zero-knowledge proofs so that data fragmentation, access control, and interoperability can be handled in a systematic and auditable way; patient and authority identities are managed with self-sovereign identity and verifiable credentials, while ownership proofs or sensitive attributes can be demonstrated without revealing full identity using zero-knowledge proofs to uphold minimal disclosure [5], [28]-[31]; clinical data are represented as FHIR resources with patient as the key entity so that each observation, encounter, or prescription carries a consistent identity reference across systems, FHIR artifacts are stored in encrypted form in off-chain storage such as IPFS, and the blockchain records only hashes, content addresses, and access-policy pointers to preserve integrity, provenance, and an audit trail without exposing record contents [1], [3], [10], [15]-[17], [32]-[34]; access is governed by patient-centric smart contracts that enforce attribute-based policies and purpose- and time-bound dynamic consent, enabling patients to grant, constrain, or revoke permissions, providing a fully logged emergency mode, and issuing third-party-verifiable proofs of access without revealing patient data through proof-based cryptography [13], [14], [35]-[37]; Polygon is selected for EVM compatibility and low transaction costs for policy operations and metadata logging, with prior work indicating efficient smartcontract use at modest average costs suitable for day-to-day healthcare operations [24], [38], [39]; resilience is enhanced by validating all contracts that interact with health data through a proxy-verification layer before deployment to filter malicious contracts and preserve business-logic integrity, while reliability metrics and security risk assessments are applied in line with best practices for FHIR systems and BC-IdM evaluation frameworks in health-IoT settings [3], [8], [23]; operationally, a service provider submits a request with proofs of relevant attributes, the smart contract evaluates consent and policy and if valid issues purpose- and time-bounded access rights, and the client retrieves the encrypted FHIR payload from IPFS and decrypts it using a session key wrapped with the patient's key, ensuring that access remains under the data owner's control, interoperable across facilities, and auditable end-to-end [1], [10], [17]

The main contribution of this work is the integration of FHIR interoperability, self-sovereign identity, and encrypted off-chain IPFS storage into a smart-contract framework that enables patientcontrolled access and verifiable auditability. In this design, FHIR payloads are encrypted end-to-end before being stored off chain, while the blockchain records only hashes, encrypted pointers, and policy metadata. Decryption is performed solely by authorized clients using patient-controlled keys, and each access event is logged on chain. The smart contracts support versioned pointers and policy-based access control, while Polygon reduces operational cost and latency. A proxy-verification layer further enhances contract integrity. Zero-knowledge proofs are noted as future work for privacy-preserving attribute verification. Overall, the framework provides portable identity, patient-centric access control, and auditable data exchange across fragmented health systems.

The novelty of this work lies in the integration of FHIR-based interoperability, SSI-driven patient identity, and dual-recipient encrypted pointers recorded on the Polygon blockchain, implemented in a fully operational prototype. Unlike prior studies that remain conceptual or focus solely on data sharing, this framework delivers verifiable auditability, patient-controlled decryption, and empirical performance evaluation across a complete registration and access workflow.

2. METHOD

Methodology cover design, implementation, and evaluation across architecture mapping, HL7 FHIR modeling, policy/consent access control, cryptographic key management for encrypted pointers, Polygon smart contracts with audit/versioning, and tests of performance, reliability, accuracy, and cost.

2.1. Logic architecture

Figure 1 illustrates the system's logical architecture, which is divided into four layers: client & trust, gateway, on-chain, and off-chain. The client & trust layer manages the patient's SSI wallet, credentials, and cryptographic operations, while providers use the console to register and access patient data. The Gateway layer validates API calls and forwards standardized FHIR requests to SATUSEHAT, coordinating registration and access workflows. On-chain, Polygon smart contracts store encrypted pointers with version history and record consent metadata for auditability. Off-chain, IPFS holds the encrypted FHIR payload and SATUSEHAT provides the authoritative Patient resource. During registration, the system creates a wallet, encrypts patient identifiers, uploads the encrypted payload to IPFS, and records references on-chain. For access, the system verifies credentials, checks consent metadata, retrieves the encrypted payload, and decrypts it for authorized users. This design enables multi-provider onboarding and gives patients cryptographically verifiable control over their data. Conceptual ZKP components are included as future enhancements for privacy-preserving attribute verification.

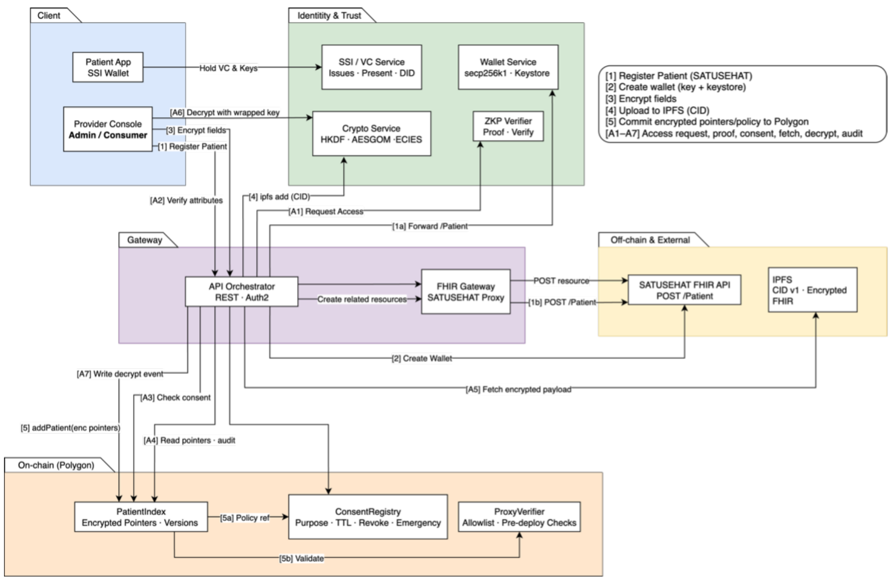
Client
Patient App SSI Wallet
Hold VC & Keys
Provider Console Admin / Consumer
[A6] Decrypt with wrapped key
[3] Encrypt fields
[1] Register Patient
Identitity & Trust
SSI / VC Service Issues · Present · DID
Wallet Service secp256k1 · Keystore
Crypto Service HKDF · AESGOM ·ECIES
ZKP Verifier Proof · Verify
[1] Register Patient (SATUSEHAT)
[2] Create wallet (key + keystore)
[3] Encrypt fields
[4] Upload to IPFS (CID)
[5] Commit encrypted pointers/policy to Polygon
[A1–A7] Access request, proof, consent, fetch, decrypt, audit
[A2] Verify attributes
[4] ipfs add (CID)
[A1] Request Access
[1a] Forward /Patient
Gateway
API Orchestrator REST · Auth2
Create related resources
FHIR Gateway SATUSEHAT Proxy
POST resource
[1b] POST /Patient
Off-chain & External
SATUSEHAT FHIR API POST /Patient
IPFS CID v1 · Encrypted FHIR
[2] Create Wallet
[A7] Write decrypt event
[A5] Fetch encrypted payload
[A3] Check consent
[5] addPatient(enc pointers)
[A4] Read pointers · audit
On-chain (Polygon)
PatientIndex Encrypted Pointers · Versions
[5a] Policy ref
ConsentRegistry Purpose · TTL · Revoke · Emergency
ProxyVerifier Allowlist · Pre-deploy Checks
[5b] Validate

Figure 1. The logical architecture

2.2. Data design and metadata

The data design in this study adopts the HL7 FHIR (Fast Healthcare Interoperability Resources) standard with the Indonesian Ministry of Health National profiles to maintain interoperability and schema consistency across systems. Element structures are represented with FHIRPath so that attribute locations, primitive data types, and terminology bindings can be traced deterministically. The metadata scope includes patient identity and status, terminology coding, timestamps, and administrative region extensions. All elements are prepared for schema validation, serialization, encryption, policy-based access control, and auditing, which enables safer and more scalable automation of ingestion, storage, and information exchange.

As the modeling and validation reference, Table 1 FHIR patient summarizes element paths using FHIRPath together with the primitive data types used in the Patient resource according to the national profile. This organization simplifies deterministic navigation of attribute positions (for example identity, demographics, clinical status, emergency contacts, and communication language) while ensuring type consistency during serialization, schema validation, and service-to-service mapping. Methodologically, the FHIRPath list serves as an operational reference for building input forms, defining validation rules including terminology, performing ETL transformations, and conducting integration testing, so the ingestion and data exchange pipeline is standardized and low in ambiguity.

Table 1. FHIR path mapping and data types

FHIR Path [1]	FHIR Path [2]	Type [1]	Type [2]
Patient.resourceType	Patient.maritalStatus.coding[0].system	code	uri
Patient.meta.profile[0]	Patient.maritalStatus.coding[0].code	canonical	code
Patient.identifier[0].use	Patient.maritalStatus.coding[0].display	code	string
Patient.identifier[0].system	Patient.maritalStatus.text	uri	string
Patient.identifier[0].value	Patient.multipleBirthInteger	string	integer
Patient.active	Patient.contact[0].relationship[0].coding[0].system	boolean	uri
Patient.name[0].use	Patient.contact[0].relationship[0].coding[0].code	code	code
Patient.name[0].text	Patient.contact[0].name.use	string	code
Patient.gender	Patient.contact[0].name.text	code	string
Patient.birthDate	Patient.contact[0].telecom[0].system	date	code
Patient.deceasedBoolean	Patient.contact[0].telecom[0].value	boolean	string
Patient.address[0].use	Patient.contact[0].telecom[0].use	code	code
Patient.address[0].line[0]	Patient.communication[0].language.coding[0].system	string	uri
Patient.address[0].city	Patient.communication[0].language.coding[0].code	string	code
Patient.address[0].postalCode	Patient.communication[0].language.coding[0].display	string	string
Patient.address[0].country	Patient.communication[0].language.text	code	string

For cryptographic identity and national health identity tracking, Table 2. Patient wallet and IHS number response specifies the data paths and types related to the patient wallet (wallet address, public key, keystore structure) and the IHS Number that appears in responses. This sequence supports access control and auditing: the patient wallet governs decryption and authorization keys, the keystore describes key protection mechanisms (cipher, KDF, parameters), and the IHS Number provides a unique identifier within the health system. The table therefore guides implementation for binding on-chain and off-chain identities, recording consent, and tracing transaction and access events, without repeating the Patient structure already covered in the previous table.

Table 2. Patient wallet keystore path and data types

Path [1]	Path [2]	Type [1]	Type [2]
response.id	patientWallet.keystoreJson.Crypto.kdfparams.dklen	integer	string
patientWallet.address	patientWallet.keystoreJson.Crypto.kdfparams.p	integer	string
patientWallet.publicKey	patientWallet.keystoreJson.Crypto.kdfparams.r	integer	hex (secp256k1 compressed)
patientWallet.keystoreJson.address	patientWallet.keystoreJson.Crypto.mac	hex	hex (lowercased)
patientWallet.keystoreJson.id	patientWallet.keystoreJson.x-ethers.client	string	uuid
patientWallet.keystoreJson.version	patientWallet.keystoreJson.x-ethers.gethFilename	string	integer
patientWallet.keystoreJson.Crypto.cipher	patientWallet.keystoreJson.x-ethers.path	string	string
patientWallet.keystoreJson.Crypto.cipherparams.iv	patientWallet.keystoreJson.x-ethers.locale	string	hex
patientWallet.keystoreJson.Crypto.ciphertext	patientWallet.keystoreJson.x-ethers.mnemonicCounter	hex	hex
patientWallet.keystoreJson.Crypto.kdf	patientWallet.keystoreJson.x-ethers.mnemonicCiphertext	hex	string
patientWallet.keystoreJson.Crypto.kdfparams.salt	patientWallet.keystoreJson.x-ethers.version	string	hex
patientWallet.keystoreJson.Crypto.kdfparams.n	patientWallet.keystorePassword	string	integer

2.3. Roles and access control

In this model, provider administrators register patients in bulk, and the system generates a crypto wallet and JSON keystore for each patient. These are delivered securely to the patient and not retained by administrators. Patients verify ownership by supplying the keystore and password, which allows a temporary local key decryption and a standard challenge–response signature. The verified wallet is then linked to the patient's DID or verifiable credential and to the corresponding FHIR Patient resource, while administrators operate only on metadata and encrypted pointers—not private keys. Access control follows policy-based evaluation using purpose, time, and attribute constraints. When clinicians request access, the Orchestrator verifies credentials, checks consent metadata, and enforces facility scoping. If approved, the system returns encrypted pointers, and the FHIR data are decrypted only by authorized key holders. Policy updates, revocation, and audit logging are supported, and each decryption event is recorded on-chain for accountability. Conceptual ZKP components are reserved for future work.

2.4. Cryptography and key management

Building on the cryptography and key-management architecture (Figure 2 cryptography and key management), the flow begins at the key holders. The system generates a patient wallet using `secp256k1` and packages the private key into a JSON keystore protected by a random Base64 password. The administrator's symmetric key version is stored in a key version registry to support key rotation. In the entropy & secrets stage, the keystore password, K_{admin} (32 bytes), and other random values such as 96-bit nonces or initialization vectors (IVs) are produced by a cryptographically secure pseudorandom number generator (CSPRNG). In the crypto service stage, HKDF-SHA256 derives two content keys, K_{CID} and K_{IHS}, from a random base key K_{rec}. Each plaintext, namely the CID (IPFS pointer) and the IHS or patient identifier, is encrypted using AES-256-GCM with associated authenticated data (AAD) that binds the recordId andwalletAddress, and for the administrator also includes the key version. The content keys are then wrapped for two recipients: for the administrator, AES-GCM is used with K_{admin} and AAD carrying the key version; for the patient, ECIES over `secp256k1` is used with the patient's public key. The results are packaged as `encCid` and `encIhs` objects containing {alg, iv, ciphertext, recipients, aad}. In the packaging & persistence stage, the encrypted FHIR payload is stored off-chain in IPFS, producing a CID, while `encCid` and `encIhs` are stored on-chain in the PatientIndex contract on Polygon as encrypted pointers together with their recipient lists. During access, two decryption paths are available. The administrator unwraps the content key using K_{admin} based on the key version and decrypts the data with AES-GCM. The patient unlocks the keystore using the password to obtain the private key, performs ECIES unwrapping to recover the content key, and decrypts the data using AES-GCM. Each successful decryption triggers an on-chain audit event, DecryptEvent, which records the recordId, the executor's address, and the timestamp.

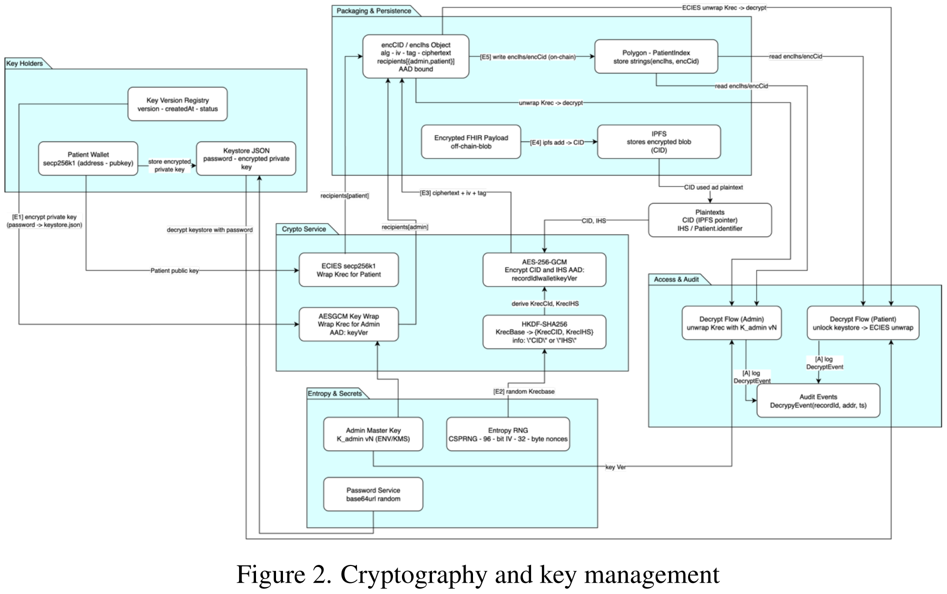

Figure 2. Cryptography and key management

2.5. Smart contract framework

Figure 3 shows the smart contract overview. The PatientIPFSManager contract deployed on Polygon (EVM) stores only verified IPFS pointers rather than clinical data. These pointers are maintained through a registry implemented as a `mapping<string, PatientPointer[]>`, which is versioned per patient identifier. The PatientPointer struct contains the CID, patient type, identifier, timestamp, createdBy, and activation status. The PatientType enumeration (`WITH_NIK`, `WITHOUT_NIK`, `BABY_NEWBORN`) is used to classify the identity source associated with each record. State updates are protected by the `onlyOwner` modifier, ensuring that all data-mutating operations can be executed exclusively by the contract owner. Core contract functions include `addPatient(...)` to append a new versioned record, `deactivatePatient(...)` to disable an existing entry, and query functions such as `getLatestPatient(...)`, `getPatientByVersion(...)`, and `getHistoryCount(...)` to support retrieval and historical inspection of patient pointers. For on-chain auditing purposes, the `PatientAdded` and `PatientDeactivated` events record essential key parameters, including the identifier, CID, patient type, creator address, timestamp, and status. This contract design preserves data integrity, traceability, and minimal on-chain storage, where the blockchain maintains only cryptographic proofs and immutable change history, while the actual clinical content remains stored off-chain in IPFS. Consequently, the architecture aligns with methodological requirements for implementing a secure and verifiable change log.

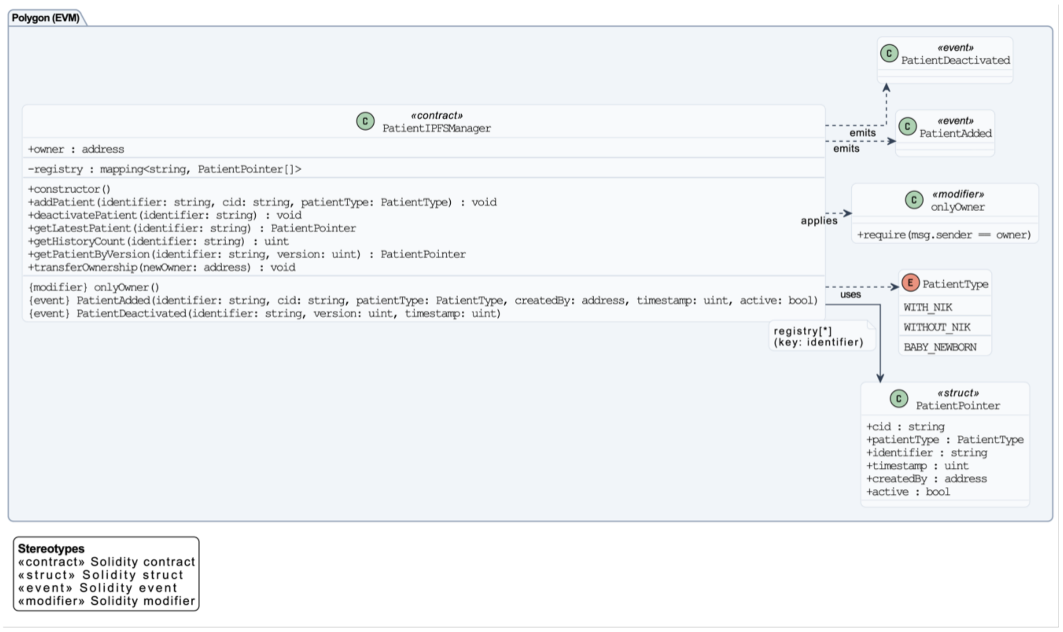
Polygon (EVM)
«event»
PatientDeactivated
«contract»
PatientIPFSManager
+owner : address
-registry : mapping<string, PatientPointer[]>
+constructor()
+addPatient(identifier: string, cid: string, patientType: PatientType) : void
+deactivatePatient(identifier: string) : void
+getLatestPatient(identifier: string) : PatientPointer
+getHistoryCount(identifier: string) : uint
+getPatientByVersion(identifier: string, version: uint) : PatientPointer
+transferOwnership(newOwner: address) : void
{modifier} onlyOwner()
{event} PatientAdded(identifier: string, cid: string, patientType: PatientType, createdBy: address, timestamp: uint, active: bool)
{event} PatientDeactivated(identifier: string, version: uint, timestamp: uint)
emits
emits
«event»
PatientAdded
applies
«modifier»
onlyOwner
+require(msg.sender == owner)
uses
PatientType
WITH_NIK
WITHOUT_NIK
BABY_NEWBORN
registry[*]
(key: identifier)
«struct»
PatientPointer
+cid : string
+patientType : PatientType
+identifier : string
+timestamp : uint
+createdBy : address
+active : bool
Stereotypes
«contract» Solidity contract
«struct» Solidity struct
«event» Solidity event
«modifier» Solidity modifier

Figure 3. Smart contract overview

For data addition, Algorithm 1 add patient describes the flow when the contract owner (onlyOwner) writes a new IPFS pointer for a patient. The algorithm constructs a PatientPointer object containing cid, patientType, identifier, timestamp, and createdBy, marks it as active = true, and appends it to registry[identifier] as the latest version. The PatientAdded event is emitted for on-chain audit, so every ingest is recorded with key parameters and remains traceable. This mechanism enforces version integrity, author traceability, and the separation of medical content (off chain) from historical proofs (on chain) within a secure and minimalist method. For logical access termination, Algorithm 2 deactivate patient presents the procedure for soft deactivation of the most recent patient entry. Only the contract owner is authorized to execute the operation. The algorithm verifies that history exists, confirms the entry has not already been deactivated, and then sets active to false. The patient deactivated event is recorded together with the version index and timestamp so that the change log can be audited. This soft deactivation preserves version records without deleting history and enables safer access policies and recovery paths.

Algorithm 1 Contract add patient

```
Data: identifier: string, cid: string, patientType: PatientType, caller: address
Result: A new PatientPointer is appended to registry[identifier] and event PatientAdded is emitted (state active = true).
 1: Begin
 2: Require caller == owner; otherwise reject as unauthorized
 3: Construct ptr ← PatientPointer{cid, patientType, identifier, timestamp = now, createdBy = caller, active = true}
 4: Append ptr to registry[identifier]
 5: Emit event PatientAdded(identifier, cid, patientType, caller, now, true)
 6: return success
 7: End
```

Algorithm 2 Contract deactivate patient

```
Data: identifier: string, caller: address
Result: Latest pointer for identifier is marked inactive; event PatientDeactivated is emitted.
 1: Begin
 2: Require caller == owner; otherwise reject as unauthorized
 3: Let $len$ ← length(registry[identifier]); if $len == 0$, reject (no record)
 4: Let $ptr$ ← registry[identifier][$len - 1$]
 5: if $ptr.active == false$ then
 6:     reject (already deactivated)
 7: end if
 8: Set $ptr.active \leftarrow false$
 9: Emit event PatientDeactivated(identifier, version = $len - 1$, timestamp = now)
10: return success
11: End
```

For read and administrative needs, Algorithm 3 query and ownership operations summarizes three read-only operations (getLatestPatient, getHistoryCount, and getPatientByVersion) and one administrative operation (transferOwnership). The algorithm retrieves versions deterministically with validity checks (for example, it rejects empty histories or invalid version indices), while ownership transfer requires the caller to be the current owner and newOwner to be a nonzero address. Together, these operations provide a consistent observation path over versioned data and strict contract governance to maintain integrity and accountability during the methodological phase.

Algorithm 3 Read and ownership operations

```
Data: identifier: string, caller: address
Result: Latest pointer for identifier is marked inactive; event PatientDeactivated is emitted.
 1: Begin
 2: Switch on op:
 3:   Case GetLatest:
 4:     Let $len$ ← length(registry[identifier]); if $len == 0$, reject (no record)
 5:     return registry[identifier][$len - 1$]
 6:   Case GetHistoryCount:
 7:     return length(registry[identifier])
 8:   Case GetByVersion:
 9:     If $version \geq$ length(registry[identifier]), reject (invalid version)
10:     return registry[identifier][version]
11:   Case TransferOwnership:
12:     Require caller == owner; otherwise reject as unauthorized
13:     Require newOwner $\neq$ address(0); otherwise reject (invalid address)
14:     Set $owner$ ← newOwner
15:     return success
16:   Default: reject (unknown operation)
17: End
```

2.6. Implementation setup

The system is implemented entirely as a web-based application using a single Next.js codebase that serves both the user interface and the API layer through route handlers in `app/api`. Smart contracts are deployed using the Thirdweb deployment framework, and the contract address is stored as the `NEXT_PUBLIC_CONTRACT_ADDRESS` environment variable. All configuration parameters are loaded through the `.env.local` file, including the Polygon network configuration (`NEXT_PUBLIC_CHAIN_ID`, `NEXT_PUBLIC_RPC_URL`), SATUSEHAT credentials (`SATUSEHAT_CLIENT_ID`, `SATUSEHAT_CLIENT_SECRET`, SA-

`TUSEHAT_URL`), IPFS endpoints (`IPFS_API_URL`, `GATEWAY_URL`), a 32-byte base encryption key (`ENC_MASTER_KEY_BASE64`), and the `ADMIN_PRIVATE_KEY`, which is accessible exclusively within the server-side execution context. On the client side, the Provider Console is used by healthcare provider administrators to register multiple patients, while the Patient Portal enables patients to import a JSON keystore and enter a keystore password for on-device verification and decryption.

The implementation relies on two primary Next.js APIs. First, the POST `/api/register-patient` endpoint forwards the request body to the SATUSEHAT `/Patient` API. Upon receiving the SATUSEHAT response, the system generates a `secp256k1` wallet for the patient, creates a JSON keystore protected by a randomly generated password, assembles a combined payload consisting of the original request, the SATUSEHAT response, and a patientWallet block, and uploads the encrypted payload to IPFS to obtain a CID. Subsequently, the system performs separate encryption for the IHS and CID using HKDF-SHA256 for key derivation and AES-256-GCM for authenticated encryption, with key wrapping applied for two recipients (administrator and patient) using AES-GCM and ECIES over `secp256k1`, respectively. The resulting `encIhs` and `encCid` objects are then submitted to the smart contract via the Thirdweb SDK to be processed by the `addPatient` function. Second, the POST `/api/request-access` endpoint verifies authorization, retrieves encrypted pointers from the blockchain, and fetches the encrypted payload from IPFS. The system then performs client-side decryption using the patient's JSON keystore and the keystore password that was previously provided by the administrator, enabling the recovery of the content key and the subsequent decryption of the payload.During development, the system utilizes the Cardano or Amoy test networks. In production, all secrets are stored as server-only environment variables, all connections are secured using HTTPS, Cross-Origin Resource Sharing (CORS) policies are strictly enforced, and request rates are limited to mitigate abuse. End-to-end testing confirms that patient registration, IPFS upload, encryption, on-chain commitment, access requests, and decryption execute consistently without requiring a separate backend component.

2.7. Test scenarios and metrics

Test scenarios and quantitative metrics are defined to evaluate the design's performance across six areas: end-to-end (E2E) flow, external integrations (SATUSEHAT and IPFS), cryptography (encryption and decryption), identity and keys (wallet and keystore), and on-chain operations (Polygon). All metrics address three evaluation objectives: efficiency and reliability of service flows (1–6); efficiency and correctness of cryptography on the client and recipient sides, covering key derivation, encryption, key wrapping or unwrapping, decryption, and accuracy (7–9); and identity overhead together with on-chain latency and transaction costs that affect operational feasibility (10–11). Measurements are taken directly at execution Points are measured in code and summarized using the mean μ and the 95th percentile ($p95$) to capture both efficiency and robustness against outliers.

First, E2E patient registration functionality is measured by total latency from the start of processing to the final response:

$$\mathrm{E2E}_i = t_i^{\mathrm{resp}} - t_i^0 \tag{1}$$

The mean and percentile are computed as:

$$\mu_{\mathrm{E2E}} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{E2E}_i, \qquad p95_{\mathrm{E2E}} = \inf\{x : F_{\mathrm{E2E}}(x) \geq 0.95\} \tag{2}$$

In addition, the end-to-end success rate evaluates process reliability:

$$\mathrm{Succ}_{\mathrm{E2E}}(\%) = \frac{N_{\mathrm{success}}}{N_{\mathrm{total}}} \times 100 \tag{3}$$

where t_i^0 denotes the handler start timestamp for trial i, t_i^{resp} is the timestamp immediately before sending the response, n is the number of trials, $F_{\mathrm{E2E}}(\cdot)$ denotes the empirical distribution function, N_{success} is the number of flows that completed without error, and N_{total} is the total number of trials.

Second, external integrations are evaluated separately. Request latency to SATUSEHAT is defined as

$$\mathrm{LAT}_{\mathrm{SATU},i} = t_i^{\mathrm{doneSATU}} - t_i^{\mathrm{callSATU}}, \qquad \mathrm{Succ}_{\mathrm{SATU}}(\%) = \frac{N_{2xx}}{N_{\mathrm{SATU}}} \times 100 \tag{4}$$

While IPFS upload performance is evaluated as

$$\mathrm{LAT}_{\mathrm{IPFS},i} = t_i^{\mathrm{doneIPFS}} - t_i^{\mathrm{callIPFS}}, \qquad \mathrm{Succ}_{\mathrm{CID}}(\%) = \frac{N_{\mathrm{validCID}}}{N_{\mathrm{uploads}}} \times 100 \tag{5}$$

The uploaded payload size is also recorded as

$$\mathrm{SIZE}_{\mathrm{payload},i} = \mathrm{byteLen}\big(\mathrm{JSON}(\mathrm{payload}_i)\big) \tag{6}$$

where t^{call} and t^{done} denote the timestamps immediately before and after the fetch call, N_{2xx} counts HTTP 2xx responses from SATUSEHAT, N_{SATU} is the number of SATUSEHAT calls, N_{validCID} is the number of IPFS uploads that produced a valid CID, N_{uploads} is the total number of uploads, and $\mathrm{byteLen}(\cdot)$ denotes the byte length of the JSON string.

Third, cryptography and encryption are evaluated by measuring HKDF key derivation for each label $\ell \in \{\mathrm{CID}, \mathrm{IHS}\}$, AES-256-GCM encryption per field, and key wrapping operations for both the administrator and the patient. The sizes of the resulting encrypted objects are then recorded.

$$\begin{aligned} T^{l}_{\mathrm{HKDF},i} &= t_{i,l}^{\mathrm{hkdf,done}} - t_{i,l}^{\mathrm{hkdf,start}}, \\ T_{\mathrm{AESenc},i} &= t_i^{\mathrm{enc,done}} - t_i^{\mathrm{enc,start}}, \\ T_{\mathrm{wrapA},i} &= t_i^{\mathrm{wrapA,done}} - t_i^{\mathrm{wrapA,start}}, \\ T_{\mathrm{wrapP},i} &= t_i^{\mathrm{wrapP,done}} - t_i^{\mathrm{wrapP,start}}. \end{aligned} \tag{7}$$

$$\mathrm{SIZE}_{\mathrm{encCID},i} = \mathrm{byteLen}\big(\mathrm{JSON}(\mathrm{encCid}_i)\big), \qquad \mathrm{SIZE}_{\mathrm{encIHS},i} = \mathrm{byteLen}\big(\mathrm{JSON}(\mathrm{encIhs}_i)\big) \tag{8}$$

Fourth, cryptography: decryption. Measure content key unwrapping for administrator and AESGCM decryption per field, then compute accuracy:

$$\begin{aligned} T_{\mathrm{unwrapA},i} &= t_i^{\mathrm{unwrap,done}} - t_i^{\mathrm{unwrap,start}}, \\ T_{\mathrm{AESdec},i} &= t_i^{\mathrm{dec,done}} - t_i^{\mathrm{dec,start}}, \\ \mathrm{ACC}_{\mathrm{dec}}\ (\%) &= \frac{N_{\mathrm{match}}}{N_{\mathrm{tests}}} \times 100, \end{aligned} \tag{9}$$

with a match if $\widehat{\mathrm{CID}} = \mathrm{CID}_{\mathrm{IPFS}}$ and $\widehat{\mathrm{IHS}} = \mathrm{IHS}_{\mathrm{SATUSEHAT}}$.
Fifth, identity and keys. Measure wallet creation, keystore creation, and keystore size:

$$\begin{aligned} T_{\mathrm{wallet},i} &= t_i^{\mathrm{w,done}} - t_i^{\mathrm{w,start}}, \\ T_{\mathrm{keystore},i} &= t_i^{\mathrm{ks,done}} - t_i^{\mathrm{ks,start}}, \\ \mathrm{SIZE}_{\mathrm{keystore},i} &= \mathrm{byteLen}(\mathrm{keystoreJson}_i). \end{aligned} \tag{10}$$

$$\begin{aligned} \mathrm{LAT}_{\mathrm{tx},i} &= t_i^{\mathrm{hash}} - t_i^{\mathrm{prep}}, \\ \mathrm{Cost}^{\mathrm{ETH}}_{\mathrm{tx},i} &= \mathrm{gasUsed}_i \times \mathrm{effectiveGasPrice}_i, \\ \mathrm{Cost}^{\mathrm{Fiat}}_{\mathrm{tx},i} &= \mathrm{Cost}^{\mathrm{ETH}}_{\mathrm{tx},i} \times P_{\mathrm{ETH}}. \end{aligned} \tag{11}$$

3. RESULTS AND DISCUSSION

Empirical results and discussion for the smart contract framework for patient identity are presented, covering cryptographic artifacts, the keystore verification workflow, end-to-end and per-stage latency distributions, on-chain cost and gas characteristics, and a statistical summary from 50 registration cases, aligned with the test scenarios and metrics specified in Methods.

Table 3 shows that both pointers, namely the identifier (IHS) and the *cid* (IPFS address), use version 1 (`v1`) encryption based on AES-256-GCM combined with ECIES over `secp256k1`. Each encrypted object includes a unique 96-bit initialization vector (IV) and a 128-bit GCM authentication tag, along with identical associated authenticated data (AAD) that binds the ciphertext to the *recordId* and *walletAddress*. Each object contains two wrapped keys for distinct recipients: an administrator wrap encrypted using AES-256-GCM with *keyVersion 1*, which protects a Base64-encoded record key (K_{rec}) to support controlled key rotation, and a patient wrap that encrypts the same K_{rec} using the patient's `secp256k1` public key derived from the *pubKeyHash*. Per-field content keys, $K_{\text{rec}}^{\text{CID}}$ and $K_{\text{rec}}^{\text{IHS}}$, isolate compromise risk such that disclosure of one pointer does not reveal the other. The encrypted artifacts are stored on-chain as compact JSON objects containing the Base64-encoded IV, authentication tag, ciphertext, wrapped keys, and AAD. This representation remains ABI-friendly and audit-ready. Successful decryption requires a valid wrapped key for either recipient, matching AAD, and a verified GCM authentication tag, thereby ensuring confidentiality, integrity, auditability, patient-centric access control, and readiness for master-key rotation without requiring re-encryption of historical records.

Table 3. Encryption artifact summary (encIhs & encCid): AES-256-GCM + ECIES

Object	`identifier`	`cid`
alg	`v1(aes-256-gcm+ecies-secp256k1)`	`v1(aes-256-gcm+ecies-secp256k1)`
cipher.iv	`a2ROJi3SqqNqakMW`	`WBQj/gMmsCvYkgx1`
cipher.tag	`biR+5au15iNSdIhEIJL/VA==`	`s91GE5AQujoEi2TLML3wYg==`
cipher.ciphertext	`HcN/PgDdwt9QfHWT`	`bxxvGMKpYJ5DS7Pn0dYYg8FyqZwkA3/xC2+8 TOSPOL0kgy8Mo88zixZudkV7ekSdztbvSHBf 0n7XyN8=`
aad	`UDIwMzk1MzU0OTY4fDB4YmZDRmZkOWU1NmY2 MUVjNDNhRkZhMjgyMzU4NTVjRWM2MTUwMTMz Qg==`	`UDIwMzk1MzU0OTY4fDB4YmZDRmZkOWU1NmY2 MUVjNDNhRkZhMjgyMzU4NTVjRWM2MTUwMTMz Qg==`
admin-aes (keyVersion, iv, tag, ciphertext)	`kv=1,iv=eyfC0PwCZsrQDE0j,tag=3Bz0VkL ISMCinButBZ53vg==,ct=Qb4rmodoRbLaJLU PF+2AJ1U4g7zBlhvIT9np0Qbe9PGc3fOa393 xksvTcec=`	`kv=1,iv=jwlmbI7QCRZRYOPR,tag=tKmrXJH WEKM/s4pm+ijRxA==,ct=cNCG5j6KMjhqDPp I5KVzGxD2kjm6cJpADDu3S2O+YDdjZkNYRpC 7I2+G6WM=`
patient-ecies (pubKeyHash, payload)	`0x5ba2d6bb...f08ba,payload=d9f1e0a80 6f40d91...12b29480`	`0x5ba2d6bb...f08ba,payload=211ceb822 a75c3a5...35b1c7c`

Figure 4 shows the end-to-end time distribution: most flows finish within 4 to 7 seconds, the median is about 6 seconds, the 95th percentile is about 7.5 seconds, and a single outlier around 13 seconds suggests sporadic spikes from network variability, external service responses, or on-chain confirmation delays. Consistent with this, Figure 5 reports average stage contributions to E2E, where the on-chain path is the dominant bottleneck at roughly two thirds of the total time, followed by SATUSEHAT at about 17 percent, wallet creation at about 12 percent, IPFS at about 3 percent, and encryption near zero. Stage-wise boxplots confirm the largest spread for on-chain time with a long upper tail, SATUSEHAT shows moderate variance with a few outliers, and IPFS and encryption are small and stable. The practical implication is to focus improvements on the on-chain path through priority-fee tuning, RPC optimization, or batching, and to reduce SATUSEHAT and wallet-creation latency; cryptographic components are not performance limiting.

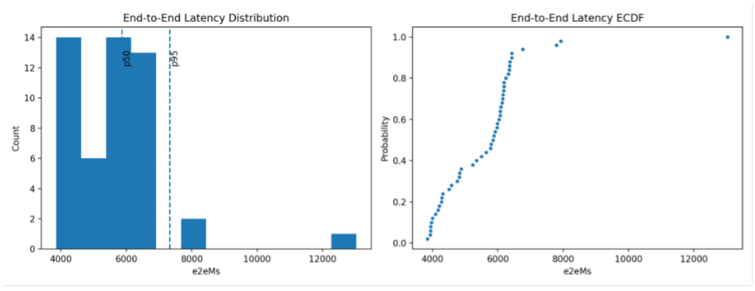

Figure 4. End-to-end latency histogram and ECDF

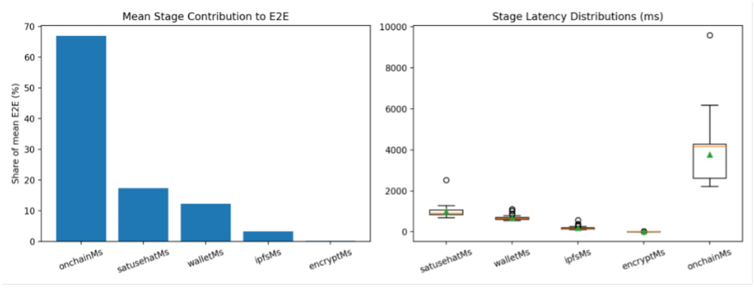

Figure 5. Mean stage contribution to E2E (bar) and per-stage latency boxplots

Figure 6 shows that transaction cost varies only slightly and has weak correlation with on-chain latency, indicating that delays are driven mainly by network and block inclusion rather than fees. Encryption time is nearly constant because payload size is fixed. Figure 7 similarly shows consistent gas usage 1,336,763 and effective gas price centers near 30 Gwei, confirming stable contract execution costs. Overall, the results suggest predictable encryption overhead, steady gas consumption.

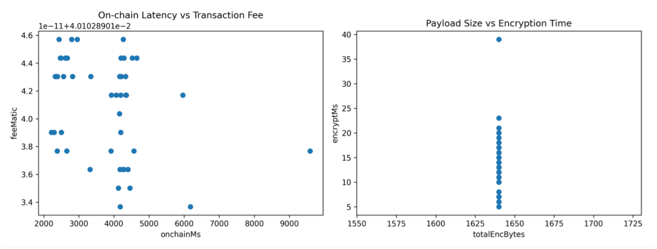

Figure 6. On-chain latency vs fee scatter, and payload size vs encryption time scatter

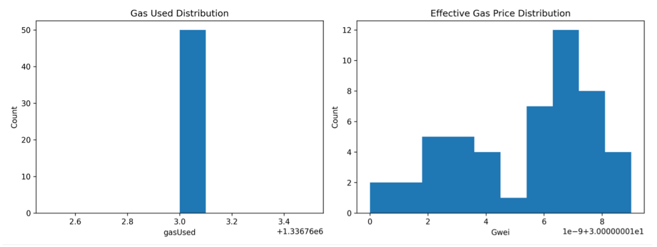

Figure 7. Gas used distribution, and effective gas price distribution in Gwei

Table 4 and Table 5 show that all 50 cases succeeded with a mined-tx ratio of 1, producing a median end-to-end time of 5.86 s (p95 7.79 s, max 13.03 s). On-chain latency dominated (median 4.16 s, mean 3.77 s, SD 1.29 s), while SATUSEHAT measured 0.914 s (mean 0.976 s), wallet creation 0.657 s (mean 0.688 s), IPFS 0.155 s (mean 0.183 s), and encryption remained lightweight at 13 ms (mean 13.9 ms, p95 20.6 ms). Encrypted artifact sizes were stable (encCid 852 bytes, encIhs 788 bytes). Across the 281.7 s run, throughput was 0.178 req/s, with an average cost of 0.0401 POL/MATIC on chain 137 for contract `0x419f837A088208F79f87d7F963a35F3030eB010`. CID and IHS decryption validation reached 100 percent.

Table 4. Descriptive statistics

Metric	Min	P50	P95	P99	Max	Mean	Std
e2eMs	3852	5870.5	7330.3	10528.51	13028	5633.18	1476.463531
satusehatMs	699	914	1213.85	1916.87	2523	976.24	267.730413
walletMs	557	657	955.5	1094.59	1099	688.10	118.575774
ipfsMs	109	154.5	338.00	480.76	567	182.78	81.274865
encryptMs	5	13	20.55	31.16	39	13.88	5.328552
onchainMs	2215	4158	5373.95	7920.14	9593	3771.44	1287.047882

Table 5. Summary statistics

Metric	Value	Metric	Value
total_cases	50	success_ratio	1
throughput_rps	0.1775101713	e2e_ms_p50	5857
e2e_ms_p95	7792	e2e_ms_p99	13028
avg_satusehat_ms	976.24	avg_ipfs_ms	182.78
avg_encrypt_ms	13.88	avg_onchain_ms	3771.44
avg_encCid_bytes	852	avg_encIhs_bytes	788
tx_mined_ratio	1	avg_fee_matic	0.04010289014
decrypt_valid_cid_ratio	1	decrypt_valid_ihs_ratio	1
duration_total_s	281.674	chain_id	137
contract			`0x419f837A088208F79f876d7F963a35F3030eB010`

4. CONCLUSION

The study demonstrates that a smart contract framework combining FHIR, SSI, end-to-end encryption, and off-chain storage on IPFS over Polygon can deliver portable, verifiable, and patientcontrolled identities. These results support the feasibility of the architecture for reducing reliance on EHR silos and strengthening access auditability. However, this study is limited by the exclusive use of synthetic patient data and the absence of field testing in real healthcare environments. Key limitations are the dominance of on-chain latency, dependence on external services, and an evaluation based on test data.

Recommended next steps include gas optimization and batching, adoption of lower-cost Layer 2 networks, account abstraction to improve user experience, refinement of ZKP proofs for attribute verification without data leakage, operational hardening for IPFS, and guided pilots in real healthcare facilities with security audits and formal contract verification. The demonstrated gains in identity portability and access auditability underscore the framework's practical relevance for improving digital health interoperability. Future enhancements particularly batching, gas-efficiency strategies, and migration to lower-cost Layer2 networks will be critical for achieving operational scalability. These advancements can position the architecture as a robust foundation for next-generation patient identity management in real clinical environments.

ACKNOWLEDGMENTS

We gratefully acknowledge the financial support from the DRTPM of the Ministry of Higher Education, Science, and Technology of the Republic of Indonesia. We also appreciate Telkom University for providing facilities, computing resources, and administrative support throughout this study.

FUNDING INFORMATION

This research was funded by the DRTPM of the Ministry of Higher Education, Science, and Technology of the Republic of Indonesia under Master Contract No. 125/C3/DT.05.OO/PL/2025. Sub-grant agreements were issued as Contract No. 7925/LL4/PG/2025 (dated 04 June 2025) and Contract No. 055/LIT07/PPM-LIT/2025 (dated 05 June 2025).

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Cahyo Prihantoro	✓	✓		✓	✓	✓			✓	✓		✓		✓
Dany Candra Febrianto	✓	✓						✓			✓		✓	
Maie Istighosah	✓	✓		✓	✓		✓		✓					
Ahmad Uffi Lestrasi Ma'ruf			✓	✓			✓							
Angger Taufiqur Rohman Winarno			✓	✓			✓							
Yudha Islami Sulistya			✓	✓			✓			✓				

C	: **C**onceptualization	I	: **I**nvestigation	Vi	: **Vi**sualization
M	: **M**ethodology	R	: **R**esources	Su	: **Su**pervision
So	: **So**ftware	D	: **D**ata Curation	P	: **P**roject Administration
Va	: **Va**lidation	O	: Writing - **O**riginal Draft	Fu	: **Fu**nding Acquisition
Fo	: **Fo**rmal Analysis	E	: Writing - Review & **E**diting		

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] A. Yazdinejad *et al.* "A review on security of smart farming and precision agriculture: Security aspects, attacks, threats and countermeasures," *Applied Sciences*, vol. 11, no. 16, p. 7518, 2021.

[2] M. R. Ahmed, A. K. M. M. Islam, S. Shatabda and S. Islam, "Blockchain-Based Identity Management System and Self-Sovereign Identity Ecosystem: A Comprehensive Survey," in *IEEE Access*, vol. 10, pp. 113436-113481, 2022, doi: 10.1109/ACCESS.2022.3216643.

[3] T. Pope, A. Patooghy and A. Sarrafzadeh, "On the Trustworthiness of FHIR-Based Internet-of-Things Digital Health Systems," *2023 IEEE 66th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Tempe, AZ, USA, 2023, pp. 279-283, doi: 10.1109/MWSCAS57524.2023.10406028.

[4] B. S. Samantray and K. H. K. Reddy, "Blockchain enabled secured, smart healthcare system for smart cities: A systematic review on architecture, technology, and service management," *Cluster Computing*, vol. 27, no. 10, pp. 14387–14415, 2024, doi: 10.1007/s10586-024-04661-7.

[5] B. Houtan, A. S. Hafid and D. Makrakis, "A Survey on Blockchain-Based Self-Sovereign Patient Identity in Healthcare," in *IEEE Access*, vol. 8, pp. 90478-90494, 2020, doi: 10.1109/ACCESS.2020.2994090.

[6] S. K. Jena, R. C. Barik, and R. Priyadarshini, "A systematic state-of-art review on digital identity challenges with solutions using conjugation of IoT and blockchain in healthcare," *Internet of Things*, vol. 25, p. 101111, 2024, doi: 10.1016/j.iot.2024.101111.

[7] Z. Huo *et al.*, "A Preliminary Case Study of Developing a Web-Based Digital Portal for Stroke Survivors Using Synthetic Personal Health Data," *2024 IEEE 12th International Conference on Healthcare Informatics (ICHI)*, Orlando, FL, USA, 2024, pp. 550-552, doi: 10.1109/ICHI61247.2024.00083..

[8] B. Alamri, I. Richardson and K. Crowley, "Cybersecuriy Risk Management and Evaluation Framework of Blockchain Identity Management Systems in HIoT: Experts Evaluation," in *IEEE Access*, vol. 12, pp. 144652-144683, 2024, doi: 10.1109/ACCESS.2024.3468379.

[9] A. Yan, G. Wang and L. Wang, "Research on the Design of Intelligent Health Information Management Service System for Modern Health Care Community," *2024 3rd International Conference on Health Big Data and Intelligent Healthcare (ICHIH)*, Zhuhai, China, 2024, pp. 271-276, doi: 10.1109/ICHIH63459.2024.11064797.

[10] N. K. Pandit, S. Das and C. K. Panda, "A Patient-Centric EHR Management System using Ethereum Blockchain," *2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS)*, Bangalore, India, 2024, pp. 1-5, doi: 10.1109/ICITEICS61368.2024.10625459.

[11] S. Singh and S. Chakraverty, "Blockchain-enabled trust-based patient-centric electronic medical record model (TPC-EMR)," *International Journal of Information Technology*, vol. 17, no. 2, pp. 1049–1061, 2025, doi: 10.1007/s41870-024-02179-0.

[12] S. Alsofyani and A. Alelayani, "Securing Patients' Healthcare Records Using Blockchain-Based Smart Contracts," *2024 1st International Conference on Logistics (ICL)*, Jeddah, Saudi Arabia, 2024, pp. 1-15, doi: 10.1109/ICL62932.2024.10788569.

[13] U. Chelladurai and S. Pandian, "A novel blockchain-based electronic health record automation system for healthcare," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 1, pp. 693–703, 2022, doi: 10.1007/s12652-021-03163-3.

[14] U. Chelladurai, S. Pandian, and K. Ramasamy, "A blockchain based patient centric electronic health record storage and integrity management for e-Health systems," *Health Policy and Technology*, vol. 10, no. 4, p. 100513, 2021, doi: 10.1016/j.hlpt.2021.100513.

[15] P. Pawar, N. Parolia, S. Shinde, T. O. Edoh, and M. Singh, "eHealthChain—a blockchain-based personal health information management system," *Annals of Telecommunications*, vol. 77, no. 1, pp. 33–45, 2022, doi: 10.1007/s12243-021-00868-6.

[16] N. Nautiyal, P. Agarwal and S. Sharma, "Rechain: A Secured Blockchain-Based Digital Medical Health Record Management System," *2023 4th International Conference on Innovative Trends in Information Technology (ICITIIT)*, Kottayam, India, 2023, pp. 1-6, doi: 10.1109/ICITIIT57246.2023.10068707.

[17] C. E. Exceline and S. Nagarajan, "Flexible access control mechanism for cloud stored EHR using consortium blockchain," *International Journal of System Assurance Engineering and Management*, vol. 15, no. 1, pp. 503–518, 2024, doi: 10.1007/s13198-022-01791-2.

[18] A. Singh and G. Rathee, "Smart contract empowered dynamic consent: Decentralized storage and access control for healthcare applications," *Peer-to-Peer Networking and Applications*, vol. 18, no. 1, p. 40, 2025, doi: 10.1007/s12083-024-01827-3.

[19] R. D. Garcia, G. Sankar Ramachandran, R. Jurdak and J. Ueyama, "A Blockchain-based Data Governance with Privacy and Provenance: a case study for e-Prescription," *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, Shanghai, China, 2022, pp. 1-5, doi: 10.1109/ICBC54727.2022.9805545.

[20] R. D. Garcia, G. Ramachandran, and J. Ueyama, "Exploiting smart contracts in PBFT-based blockchains: A case study in medical prescription system," *Computer Networks*, vol. 211, p. 109003, 2022, doi: 10.1016/j.comnet.2022.109003.

[21] M. Indushree and M. Raj, "A novel blockchain-based authentication scheme for telecare medical information system," *The Journal of Supercomputing*, vol. 80, no. 1, 2024, doi: 10.1007/s11227-023-05526-3.

[22] M. R. Patruni and A. G. Humayun, "BeLAS: Blockchain-envisioned lightweight authentication scheme for securing eHealth records," *Peer-to-Peer Networking and Applications*, vol. 17, no. 6, pp. 4175–4196, 2024, doi: 10.1007/s12083-024-01779-8.

[23] K. Gohil, A. Gohil, R. Gupta, N. K. Jadav and S. Tanwar, "Proxy-based Verification of Smart Contracts for Healthcare Domain in Blockchain Environment," *2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Greater Noida, India, 2023, pp. 770-775, doi: 10.1109/ICCCIS60361.2023.10425319.

[24] P. Verma, V. Tripathi and B. Pant, "ZeroMedChain: Layer 2 Security and Zero-knowledge Proof Integration for Decentralized Identity and Access Management in Healthcare," *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2024, pp. 1023-1027, doi: 10.23919/INDIACom61295.2024.10498190.

[25] H. Szczepaniuk and E. K. Szczepaniuk, "Cryptographic evidence-based cybersecurity for smart healthcare systems," *Information Sciences*, vol. 649, p. 119633, 2023, doi: 10.1016/j.ins.2023.119633.

[26] S. -V. Ionescu, "E-prescription using blockchain technology," *2022 IEEE International Conference on Blockchain, Smart Healthcare and Emerging Technologies (SmartBlock4Health)*, Bucharest, Romania, 2022, pp. 1-7, doi: 10.1109/SmartBlock4Health56071.2022.10034520.

[27] A. A. Varfolomeev and L. H. Al-Farhani, "Blockchain Based Cloud Authentication Framework for Digital Identity Management in Smart City," *2023 International Conference on Information Technology, Applied Mathematics and Statistics (ICITAMS)*, Al-Qadisyia, Iraq, 2023, pp. 86-90, doi: 10.1109/ICITAMS57610.2023.10525413.

[28] T. K. Saragih, E. Tanuwijaya and G. Wang, "The Use of Blockchain for Digital Identity Management in Healthcare," *2022 10th International Conference on Cyber and IT Service Management (CITSM)*, Yogyakarta, Indonesia, 2022, pp. 1-6, doi: 10.1109/CITSM56380.2022.9935935.

[29] P. V. M, S. N. Qurashi, F. Sobia, F. Harahsheh, S. Surendran and S. S. C. Mary, "Decentralized Identity Management Using Blockchain for Healthcare Systems," *2024 IEEE Silchar Subsection Conference (SILCON 2024)*, Agartala, India, 2024, pp. 1-6, doi: 10.1109/SILCON63976.2024.10910771.

[30] O. E. Fadul, Y. Kumar, A. Garg and K. Saluja, "A Review on Blockchain-based Digital Identity Management System," *2023 International Conference on Advanced Computing & Communication Technologies (ICACCTech)*, Banur, India, 2023, pp. 713-720, doi: 10.1109/ICACCTech61146.2023.00119.

[31] M. Htet, P. T. Yee and J. R. Rajasekera, "Blockchain based Digital Identity Management System: A Case Study of Myanmar," *2020 International Conference on Advanced Information Technologies (ICAIT)*, Yangon, Myanmar, 2020, pp. 42-47, doi: 10.1109/ICAIT51105.2020.9261785.

[32] K. M, P. C and V. S. Akshaya, "Blockchain based Healthcare Data Management," *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*, Pudukkottai, India, 2022, pp. 392-396, doi: 10.1109/ICACRS55517.2022.10029059.

[33] G. S. K, R. Kalpana, M. SK, K. S and P. PS, "Health Record Management Using Blockchain Technology," *2022 8th International Conference on Smart Structures and Systems (ICSSS)*, Chennai, India, 2022, pp. 1-5, doi: 10.1109/ICSSS54381.2022.9782228.

[34] G. A. Haryadi, Allwinnaldo, J. M. Lee and D. -S. Kim, "Enhanced Healthcare System with NFT-Prescription," *2023 14th International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju Island, Korea, Republic of, 2023, pp. 942-946, doi: 10.1109/ICTC58733.2023.10392924.

[35] P. Rastogi, D. Singh, and S. S. Bedi, "An improved blockchain framework for ORAP verification and data security in healthcare," *Journal of Ambient Intelligence and Humanized Computing*, vol. 15, no. 6, pp. 2853–2868, 2024, doi: 10.1007/s12652-024-04780-4.

[36] A. P. Singh, D. Jain, P. Vats, A. Tiwari and D. Bhadana, "The Revolutionary Potential of Blockchain using Smart Contracts: Unlocking the Next Generation of Technology," *2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, India, 2023, pp. 1459-1464, doi: 10.1109/ICAC3N60023.2023.10541496.

[37] P. Marry, K. Yenumula, A. Katakam, A. Bollepally and A. Athaluri, "Blockchain based Smart Healthcare System," *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*, Coimbatore, India, 2023, pp. 1480-1484, doi: 10.1109/ICSCSS57650.2023.10169704.

[38] E. Y. Budi, C. Prihantoro, and N. E. W. Nugroho, "Design of an e-voting website using smart contracts on the Polygon blockchain (in Indonesian: *Perancangan website e-voting menggunakan smart contract pada blockchain Polygon*)," *The Indonesian Journal of Computer Science*, vol. 12, no. 3, 2023, doi: 10.33022/ijcs.v12i3.3234.

[39] D. Kumari, S. Sharma, M. Chawla, and S. Panda, "A manifesto for healthcare-based blockchain: Research directions for the future generation," *Journal of The Institution of Engineers (India): Series B*, vol. 105, no. 5, pp. 1429–1450, 2024.

BIOGRAPHIES OF AUTHORS

Cahyo Prihantoro is lecture at Department of Informatics, Telkom University Purwokerto, Indonesia. He interest a research area in Computer Engineering and Information Technology with specialization in cloud computing, network computer, blockchain and information technology. His research areas are Quality of Services (QoS), network performance, enterprise system. He is also a member of the Center of Excellence (CoE) for ICT Infrastructure, Smart Manufacture and Digital Supply Chain (ISDigit). He can be contacted at email: cahyop@telkomuniversity.ac.id.

Dany Candra Febrianto is a lecturer at Telkom University Purwokerto. He is actively involved in teaching and research, focusing on computer vision, and natural language processing and sometimes software engineer. He is also a member of the Center of Excellence (CoE) for Sustainable Cities, Villages, and Food Security (SIVIFO). He can be contacted at email: danycandra@telkomuniversity.ac.id.

Maie Istighosah is a lecturer at Telkom University Purwokerto. She is actively involved in teaching and research, focusing on image processing, computer vision, and natural language processing. She is also a member of the Center of Excellence (CoE) for Good Health and Well-being Technologies, where she contributes to advancing digital health technologies and data-driven innovations. She can be contacted at email: maieistigh@telkomuniversity.ac.id.

Ahmad Uffi Lestrasi Ma'ruf is a Software Engineering student at Telkom University Purwokerto. His academic interests focus on blockchain and software engineering, and he is actively involved in related coursework and research projects. He can be contacted at email: ahmaduffi@student.telkomuniversity.ac.id.

Angger Taufiqur Rohman Winarno is an Informatics Engineering student at Telkom University Purwokerto. His academic interests focus on blockchain and software engineering, and he is actively involved in related coursework and research activities. He can be contacted at email: anggertaufiqurrohman@student.telkomuniversity.ac.id.

Yudha Islami Sulistya is a lecturer and practitioner at Telkom University Purwokerto, specializing in Flutter development, full-stack web engineering, and machine learning. Guided by the view that technology is an art, he designs digital experiences that are not only functional but also memorable. A Master of Computer Science graduate, he continually explores the boundaries of technology, blending creativity and scientific rigor to build impactful solutions. He is also a member of the Center of Excellence (CoE) for Good Health and Well-being Technologies, contributing to advances in digital health technologies and data-driven innovations. He can be contacted at email: yudhaislami@telkomuniversity.ac.id.