Towards Compliant and Private EHR Sharing: An Experimental Evaluation of ZKP-Blockchain Integration for Healthcare Data

Yan Watequlis Syaifudin [1,*], Vipkas Al Hadid Firdaus [1], Imam Fahrur Rozi [1], Chandrasena Setiadi [2], Suryani Dyah Astuti [3], Nobuo Funabiki [4], Maskur [1], and Kadek Suarjuna Batubulan [4]

[1] Department of Information Technology, State Polytechnic of Malang, Malang, Indonesia
[2] Department of Electrical Engineering, State Polytechnic of Malang, Malang, Indonesia
[3] Department of Biomedical Engineering, Airlangga University, Surabaya, Indonesia
[4] Department of Electrical Engineering, Okayama University, Okayama, Japan
Email: qulis@polinema.ac.id (Y.W.S.); vipkas@polinema.ac.id (V.A.H.F.); imam.rozi@polinema.ac.id (I.F.R.); chandrasenasetiadi@polinema.ac.id (C.S.); suryanidyah@fst.unair.ac.id (S.D.A.); funabiki@okayama-u.ac.jp (N.F.); maskur@polinema.ac.id (M.); pzc37um1@s.okayama-u.ac.jp (K.S.B.)
*Corresponding author

***Abstract*—The digitization of health records has enhanced clinical efficiency, but amplified risks related to data privacy, integrity, and auditability. While permissioned blockchains offer immutability and traceability, they often fail to reconcile transparency with confidentiality—either exposing sensitive data or obscuring it beyond regulatory scrutiny. To address this gap, this paper presents an integrated framework that combines Zero-Knowledge Proofs (ZKPs) with a permissioned blockchain to enable verifiable yet private healthcare transactions. A visit centric Electronic Health Record (EHR) model supports three real-world use cases: medication validity, procedure confirmation, and demographic verification. A four-layer architecture decouples data, application logic, cryptographic trust, and audit logging, allowing end-to-end validation without raw data disclosure. Experimental evaluation across three ZKP libraries (snarkJS, ZoKrates, and gnark) on a synthetic dataset of 1,000 patient visits demonstrates sub-500 ms verification latency, with snarkJS selected for its ecosystem compatibility despite slower raw performance. End-to-end pipeline latency averages 1.35 s, confirming feasibility for batch workflows such as insurance claims. The system further includes a web-based auditor interface that validates tamper-evidence under off-chain attacks, bridging cryptographic guarantees with operational compliance.**

***Keywords*—privacy-preserving, zero-knowledge proof, auditability, permissioned blockchain, electronic health records**

I. Introduction

The progressive digitization of health records has transformed clinical workflows and enabled data-driven advancements in patient care [1]. Electronic Health Record (EHR) systems consolidate comprehensive patient information, including medical histories, diagnostic results, treatment plans, and personal identifiers, into centralized repositories. While this digitization improves accessibility and coordination among healthcare providers, it also introduces substantial challenges in safeguarding sensitive data against unauthorized access, breaches, and misuse [2]. The sensitive nature of health information requires robust mechanisms to ensure confidentiality, integrity, and availability, particularly in light of increasing regulatory requirements and evolving cyber threats [3].

Conventional centralized EHR architectures are inherently vulnerable to single points of failure and present attractive targets for malicious actors, as evidenced by numerous high-profile data breaches [4]. These systems often lack transparency, making it difficult to verify who has accessed patient data and for what purpose. Decentralized alternatives, such as blockchain technology, offer a paradigm shift by distributing trust across a network of participants [5]. Through cryptographic hashing, consensus mechanisms, and immutable logging, the blockchain can provide enhanced security, transparency, and auditability [6]. In healthcare contexts, permissioned blockchains are especially promising, as they balance accountability with controlled access, aligning well with the demands of the health sector [7].

Despite the potential of blockchain to improve data integrity and traceability, a significant gap remains in existing frameworks concerning the simultaneous assurance of privacy and auditability in EHR access control [8]. Many blockchain-based solutions prioritize transparency at the expense of confidentiality, potentially exposing sensitive patient information through transaction visibility [9]. In contrast, privacy-enhancing techniques often obscure data flow to such an extent that verification and compliance auditing become impractical. There is a critical need for an integrated

Manuscript received December 18, 26, 2025; revised January 20, 2026; accepted February 6, 2025; published May 22, 2026.

approach that allows healthcare organizations to cryptographically prove the validity of specific claims without disclosing the underlying private health data.

This paper introduces a framework that integrates Zero-Knowledge Proofs (ZKPs) [10] with a permissioned blockchain [11] to enable private and verifiable healthcare transactions. The system design establishes a clinical data model with a visit-centric EHR structure. Three case studies are developed for medication validity, procedure confirmation, and demographic validation. Then a four-layer architecture is developed comprising data, application, trust, and blockchain layers. The experiment setup implements a modular technology stack using PostgreSQL, Node.js, and three ZKP libraries (snarkJS, ZoKrates, gnark) evaluated on a synthetic dataset of 1,000 patient visits based on real hospital data with their ZKP circuits. Performance metrics, including proof generation time, verification latency, and proof size, are measured. Finally, blockchain smart contracts are implemented and deployed on Hyperledger Fabric to record verification metadata.

The experimental results demonstrate that the proposed framework is effective for privacy-preserving verification in healthcare, with all three evaluated ZKP libraries performing below the 500 ms threshold suitable for real-time applications. ZoKrates showed the fastest and most consistent execution times, while gnark produced the smallest proof sizes. However, snarkJS was selected for the final implementation due to its superior integration capabilities with web-based backends and strong developer ecosystem, despite its slower raw performance. The end-to-end latency measurements for the integrated pipeline, including the database query, ZKP process, and blockchain commit, averaged approximately 1.35 s, confirming the feasibility for batch-processing workflows such as insurance claims.

This paper makes four key contributions that distinguish it from existing ZKP-blockchain healthcare frameworks: (1) a visit-centric clinical data model grounded in real hospital workflows; (2) a modular four-layer architecture that cleanly separates data storage, application orchestration, ZKP-based trust, and blockchain audit layers; (3) an experimental evaluation of three state-of-the-art ZKP libraries on a dataset of 1,000 synthetic patient visits, with metrics on proof size, generation/verification time, and integration trade-offs; and (4) a fully functional, featuring a web-based auditor dashboard that demonstrates tamper-evidence even when the ledger state is altered off-chain, effectively bridging cryptographic guaranties with real-world compliance needs.

The remainder of this paper is structured as follows: Section II introduces the research background, motivation, problem statement, and objectives. Section III provides an overview of Zero-Knowledge Proofs and permissioned blockchain technology. Section IV details the proposed framework consisting of a clinical data model, ZKP use cases, and the four-layer system architecture. Section V presents the implementation of ZKP circuits and blockchain integration, including experimental setup and results. Section VI discusses the practical implications, limitations, and lessons learned. Finally, Section VII concludes the paper and outlines directions for future work.

II. Background and Related Work

This section explains the cryptographic foundations, system architecture, and related work that underpin the integration of zero-knowledge proofs with permissioned blockchain for healthcare data verification.

A. Zero-Knowledge Proof (ZKP)

A ZKP is a cryptographic protocol that allows one party (the prover) to prove to another party (the verifier) that a given statement is true, without revealing any information beyond the truth of the statement itself [12]. This property makes ZKPs a powerful tool for privacy-preserving systems, especially in sensitive domains such as healthcare, where data confidentiality must be maintained while still enabling verification [13]. Formally, a ZKP system must satisfy three fundamental properties:

- Completeness: If the statement is true, an honest prover can convince an honest verifier of its truth.
- Soundness: If the statement is false, no dishonest prover can convince the verifier that it is true, except with negligible probability.
- Zero-Knowledge: The verifier learns nothing about the secret input or witness used in the proof, other than the fact that the statement is true.

Two prominent variants of ZKPs are zero-knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARKs) and zero-knowledge Scalable Transparent ARguments of Knowledge (zk-STARKs) [14]. While both offer strong privacy guarantees, this work focuses on zk-SNARKs due to their high efficiency, small proof size, and maturity of supporting tools (e.g., snarkJS, ZoKrates, gnark). zk-SNARKs generate succinct proofs that can be verified quickly, making them suitable for integration into real-world applications [15]. However, they typically require a trusted setup phase, which must be carefully managed to prevent compromise. In contrast, zk-STARKs offer transparency (no trusted setup) and post-quantum security, but at the cost of larger proof sizes and higher computational overhead, which limits their practicality in resource-constrained environments.

In the context of verifiable healthcare transactions, there are several types of ZKP constructions to validate clinical claims over structured EHR data:

- Set-Membership Proof proves that a value x belongs to a predefined set S, i.e., $x \in S$. Example: Proving that a prescribed medication is included in the insurer's approved formulary, without disclosing the prescription.
- Range Proof proves that a numeric value x lies within a specific interval $[a, b]$, i.e., $a \leq x \leq b$. Example: Verifying that a patient's age is at least 18 years old or that the quantity of dispensed

medication does not exceed the maximum allowable dose.

- Equality Proof proves that a value x is equal to a specific constant c, i.e., $x = c$. Example: Confirming that a patient's gender is "female" for eligibility in maternal health programs, without revealing additional personal details.

These primitive proof types can be combined within arithmetic circuits to support complex, policy-based validations, such as proving that a medical procedure is covered and the patient meets demographic criteria, while preserving the privacy of all nonrelevant health data [16].

B. Permissioned Blockchain and Auditability

Unlike public blockchains, which are open to anonymous participants and rely on energy-intensive consensus mechanisms like proof-of-work, permissioned blockchains operate under a controlled environment where network participants are known and vetted [17]. Access to the network is identity-based and requires cryptographic credentials to join, read, write, or validate transactions. Validators are preauthorized entities (e.g., hospitals, regulators), eliminating the need for mining and enabling faster, more efficient consensus processes such as Practical Byzantine Fault Tolerance (PBFT) or Raft [18]. These characteristics make permissioned blockchains highly suitable for regulated sectors such as healthcare, where data privacy, compliance with regulations, and operational efficiency are critical. They enable organizations to maintain data integrity and auditability while enforcing strict access controls and accountability.

A prominent example is Hyperledger Fabric [19], which supports modular architecture through features such as:

- Channels: Isolated communication layers that allow private transactions between specific participants.
- Chaincode (Smart Contracts): Programs that encapsulate business logic that is executed within the blockchain network, such as verifying the validity of a ZKP.
- Private Data Collections: Mechanisms to store sensitive data off-chain but still enforce its confidentiality and verification within transactions.
- Consensus via Raft/PBFT: Crash-fault or Byzantine-fault tolerant agreement protocols that ensure consistency and finality without computational waste.

In this work, Hyperledger Fabric serves as the foundation for recording verification outcomes, providing a tamper-proof, auditable trail of ZKP-based validations while preserving the privacy of underlying medical data [20].

C. Privacy Challenges in EHR Systems

In recent years, significant research has focused on enhancing privacy and security in Electronic Health Records (EHR) [21]. The increasing digitization of health data has prompted several studies to explore methods to ensure confidentiality and integrity within EHR systems. However, administrative processes such as insurance claims, regulatory reporting, and inter-institutional data sharing often require third parties to verify clinical or demographic information, leading to unnecessary exposure of sensitive patient data. Full record disclosure during these processes increases the risk of data breaches, misuse, and unauthorized profiling. Regulatory frameworks such as the General Data Protection Regulation (GDPR) in the EU [22] and personal data protection law in Indonesia [23] mandate data minimization and purpose limitation, requiring organizations to collect and share only what is strictly necessary [24]. Despite these legal requirements, many current systems fail to comply due to technical limitations in implementing granular data control.

Existing privacy-preserving techniques exhibit critical shortcomings in meeting these regulatory and operational demands [25]. Although encryption ensures data confidentiality at rest and in transit, it does not support selective disclosure (once decrypted, all data becomes visible), thus violating the principle of minimal exposure [26]. Similarly, traditional anonymization techniques are increasingly ineffective, as advances in data correlation and reidentification attacks have demonstrated that supposedly "deidentified" records can often be linked back to individuals, especially in rich datasets containing demographics, diagnoses, and treatment histories [27]. Although innovations such as homomorphic encryption, differential privacy, and blockchain-based access controls have shown promise, they often lack practical mechanisms for verifiable claims without full data release [28]. This gap underscores the need for more sophisticated solutions, such as zero knowledge proofs that enable cryptographic verification of specific attributes, while preserving end-to-end privacy and supporting compliance with modern data protection standards [29, 30].

D. Related Work

Recent investigations into the application of zero-knowledge proofs in healthcare demonstrate their potential to enhance privacy and security in sensitive data management, such as genomic information and consent processes. For instance, a study on a healthcare identity management system based on Fabric blockchain and ZKP reveals how ZKP can enable secure proof of identity while preserving privacy [31]. Similarly, another research focused on dynamic consent management systems identifies severe limitations in existing frameworks and proposes ZKP as a key mechanism to effectively enforce privacy and security protocols [32]. Concurrently, novel systems employing ZKPs in conjunction with blockchain technologies are emerging, showcasing tangible benefits in areas such as consent management, where users can maintain control over their identities without compromising sensitive health data [13]. Furthermore, the implementation of smart contracts within Hyperledger enables the precise definition and enforcement of access policies, strengthening the governance of patient data interactions [33].

However, despite these promising advancements, many prior studies remain primarily theoretical or fail to integrate effectively with existing Electronic Health

Record (EHR) schemas and practical blockchain frameworks. A notable limitation is the lack of experimental validation in real-world scenarios, which restricts the practical applicability of the proposed solutions [13, 34]. Furthermore, while conceptual frameworks have been developed, they often do not address the intricacies of interoperability with established EHR systems, which is essential for widespread adoption in healthcare contexts [32]. This is where the current work makes a significant contribution. By offering an experiment-driven integration of ZKP within a permission blockchain framework tailored for healthcare transactions, it addresses gaps in the literature and pushes beyond simple theoretical propositions to a tangible operational application [34]. This approach not only enhances private and verifiable healthcare transactions but also serves as a model for future innovations at the intersection of blockchain technology and digital health solutions.

To situate this work within the broader literature, a comparative analysis with representative state-of-the-art EHR sharing frameworks has been added. Table I summarizes salient differences across design goals, privacy mechanisms, blockchain models, interoperability orientation, and implementation maturity.

TABLE I. COMPARATIVE SUMMARY OF REPRESENTATIVE EHR SHARING FRAMEWORKS

Framework	Privacy Mechanism	Blockchain Model	Interoperability	Implementation Status	Audit/Usability
MedRec [35]	Hashing + distributed pointers	Public/consortium design	API-based integration to provider DBs	Prototype/proof-of-concept.	Logging/audit oriented; limited selective disclosure.
FHIRChain [36]	Off-chain storage + access control	Permissioned emphasis; design for scalability	Explicit alignment to HL7 FHIR resources	Concept and prototype with ONC-alignment analysis.	Focus on secure exchange; auditor UX not primary.
MedChain [37]	Digest-chain, session control	Hybrid P2P + blockchain	Session-based sharing; claims for efficiency	Prototype with performance claims	Emphasizes efficient sharing; limited ZKP-style privacy.
Survey/Reviews [13, 32, 34]	Various (encapsulation, encryption, encryption,	Public and permissioned comparisons	Discuss FHIR adoption	Literature synthesis; identifies open challenges	Notes gaps in deployable privacy-preserving verification.
This work (current manuscript)	ZKPs (zk-SNARKs) + on-chain metadata	Permissioned blockchain (Hyperledger Fabric)	Designed to be data-model agnostic; adaptable to FHIR attributes	Fully implemented pipeline (PoC) with experimental benchmarking	Functional web-based auditor demonstrating tamper-evidence and non-technical usability

III. METHODOLOGICAL FRAMEWORK

This section presents the design of a methodological framework consisting of a clinical data model, ZKP-based use cases, and a four-layer architecture that establishes a foundation for private and verifiable healthcare transactions.

A. Clinical Data Model

The clinical data model is designed to capture patient information in various healthcare workflows observed in multiple hospital settings. At its core is the visit_m table, which serves as the central entity to record each patient encounter, as illustrated in Fig. 1.

This table stores key demographic and visit-specific attributes, including id_visit (a unique identifier for each visit), time_of_visit (timestamp of the visit), id_patient (medical record number), patient_name, age, sex, blood_type, occupational, and ethnicity. By linking all medical services to a specific visit, this model allows systematic tracking of patient interactions, supporting both clinical decision-making and administrative processes such as claims and reporting.

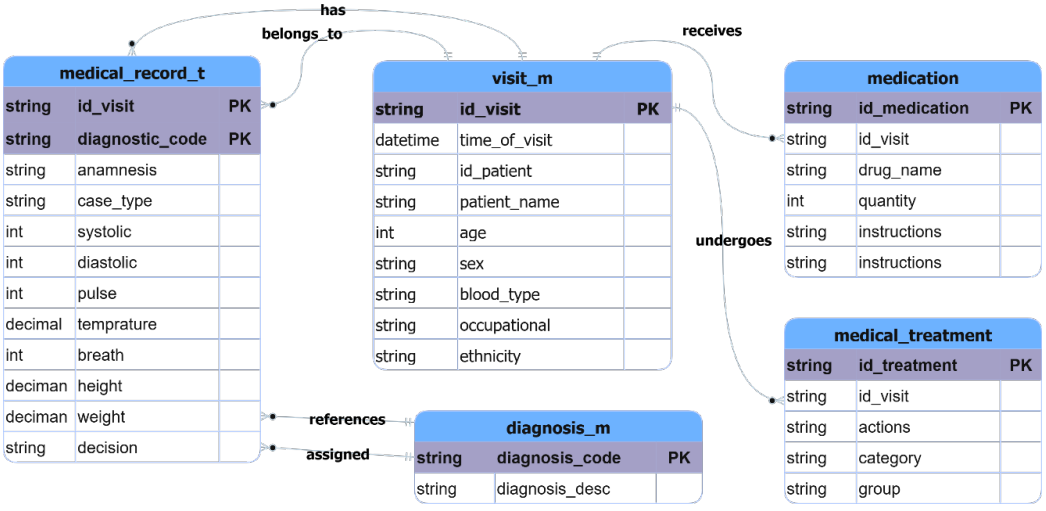

Fig. 1. Entity relationship diagram of the EHR system.

The EHR schema is custom-designed based on anonymized workflows from medium-sized hospitals but maps directly to core HL7 FHIR R4 resources [38, 39]. Specifically:

- The visit_m table corresponds to the encounter resource;
- Patient demographics align with the patient resource;
- Medication records follow the structure of MedicationRequest;
- Diagnoses and procedures are modeled after Condition and Procedure, respectively.

This semantic alignment ensures that the underlying data model is compatible with global interoperability standards, even if the current implementation uses a relational PostgreSQL backend.

B. Use Cases for ZKP-Based Verification

Based on the clinical data model and the operational workflows observed in multiple hospitals, the three key use cases are identified as presented in Table II. ZKPs provide significant privacy enhancement while enabling verifiable claims. These scenarios leverage structured data from the EHR system to generate cryptographic proofs that confirm specific attributes without exposing sensitive patient information.

TABLE II. ZKP-BASED VERIFICATION USE CASES IN HEALTHCARE

Use Case	Data Source	ZKP Types	Example Policy Constraint
Medication Validity and Quantity	medication table (drug_name, quantity)	Set-membership + Range proof	Prove: drug_name ∈ {Amlodipine, Paracetamol, …,} and quantity ≤ 30 (max allowed units)
Procedure Category Confirmation	medical_treatment table (category)	Set-membership proof	Prove: category ∈ {“laboratory”, “imaging”} (covered services)
Demographic Attribute Validation	visit_m table (age, sex, blood_type)	Range + Equality + Set-membership proofs	Prove: age ≥ 18 (adult eligibility), sex = “female” (maternal program), blood_type ∈ {A, B, AB, O}

TABLE III. ZKP CIRCUIT LOGIC MAPPING FOR HEALTHCARE USE CASES

Use Case	Public Inputs	Private Inputs (Witness)	Logical Constraints
Medication Valid.	Drug Hash, Max Threshold	Actual Drug Name, Actual Quantity	$H(drug) == H_{pub} \wedge qty \leq max$
Procedure Conf.	Category Hash	Procedure Name, Proc. Category	$Cat \in \{set\} \wedge H(proc) == H_{pub}$
Demographic Valid.	Minimum Age	Patient Birth Date, Current Date	$(Current - Birth) \geq Min\ Age$

1) Medication validity and quantity

In administrative and compliance processes, it is often necessary to verify that prescribed medications are valid and dispensed within allowable limits. Using data from the medication table (specifically, drug_name and quantity fields), a hospital can generate a ZKP to prove two conditions simultaneously:

- The prescribed drug is included in a predefined list of approved medications (drug_name ∈ approved_set),
- The quantity dispensed does not exceed a defined threshold (quantity ≤ max_allowed).

This combined proof leverages a set-membership proof for drug validity and a range proof for quantity verification. The proof ensures that only essential information is disclosed, preserving the confidentiality of other prescriptions and clinical details.

2) Procedure category confirmation

Verification of medical services often requires confirming that a procedure falls within an authorized category (e.g., laboratory tests, imaging, consultations). By referencing the category field in the medical_treatment table, a ZKP can be constructed to prove that a given medical action belongs to a permitted set of categories (e.g., category ∈ {“laboratory”, “imaging”}). This constitutes a set-membership proof, which allows auditors or regulatory bodies to validate service eligibility without accessing the full scope of medical treatments performed or associated personal health data.

3) Demographic attribute validation

For access control and eligibility validation in healthcare programs, it is often sufficient to confirm basic demographic attributes without revealing full identity. Using fields from the visit_m table (such as age, sex, and blood_type) ZKPs can be used to selectively disclose verified information. For instance:

- A range proof can confirm that age ≥ 18 for adult-specific services.
- An equality proof can verify that sex = “female” for maternal health eligibility.
- A set-membership proof can demonstrate that blood_type ∈ {A, B, AB, O} without revealing the exact type.

These proofs enable policy-compliant validation while minimizing data exposure.

To provide a precise technical specification of the privacy-preserving mechanisms utilized in these cases, we outline the circuit logic in Table III. This mapping explicitly distinguishes between the public parameters available to verifiers and the private witnesses (sensitive patient data) that remain off-chain, detailing the logical constraints enforced by the zero-knowledge circuits.

C. High-Level Architecture

The proposed system is designed to be operated by a hospital as the prover entity, enabling secure and private verification of patient data for external parties without compromising confidentiality. The architecture follows a modular four-layer design to ensure separation of concerns, scalability, and compliance with privacy

regulations, as illustrated in Fig. 2. The layered architecture shows how data flows from secure storage through proof generation to final audit logging, including:

- Data Layer: Built on PostgreSQL, this layer stores structured Electronic Health Records (EHRs) based on a defined ERD model.
- Application Layer: Implemented using Node.js, this layer orchestrates data retrieval, ZKP workflow coordination, and communication with internal and external systems.
- Trust Layer: This layer hosts the ZKP engine (snarkJS), which generates cryptographic proofs without exposing raw data. Only the proof and public inputs are shared externally.
- Blockchain Layer: A permissioned blockchain network with Hyperledger Fabric provides an immutable and auditable ledger to record verification results, ensuring accountability while preserving privacy.

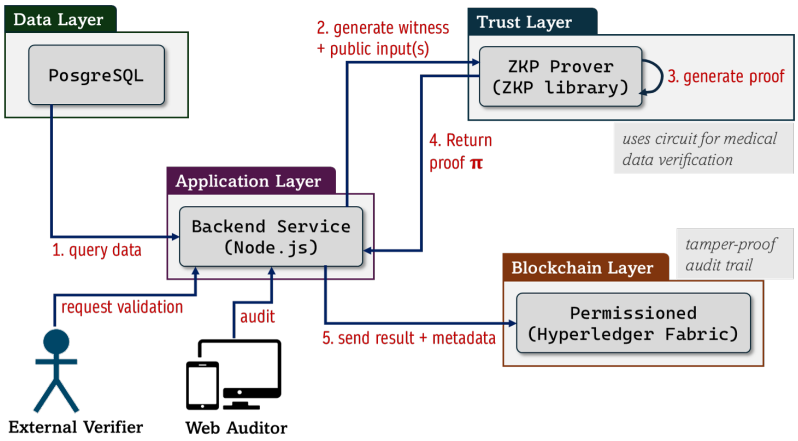

Fig. 2. High-level architecture with four layers.

The ZKP verification process follows a structured end-to-end flow:

- Request from verifier: An external party (e.g., insurance company, online pharmacy) as verifier sends a verification request to the hospital's backend (application layer), specifying the claim to be proven.
- Data extraction and witness generation: The application layer queries the PostgreSQL database (data layer) to retrieve the relevant attribute (e.g., age). This value is used as a private input (witness) for the ZKP circuit.
- Proof generation: The Trust Layer generates a cryptographic proof using a predefined circuit and a ZKP library. The proof substantiates the claim without disclosing the underlying data.
- Proof transmission to verifier: The hospital sends the generated proof π and minimal public inputs (e.g., policy threshold) to the verifier.
- Verification and acknowledgment: The verifier runs a verification algorithm using a public verification key. If the proof is valid, the verifier acknowledges receipt of the successful validation.
- Blockchain logging: Upon receiving the acknowledgment, the hospital records a metadata log on the Blockchain Layer via a smart contract. This log includes proof hash, visit ID, timestamp, and result, creating a tamper-proof audit trail.

This flow ensures that sensitive patient data remains within the hospital's trusted environment while enabling verifiable, accountable, and privacy-preserving interactions with external systems.

IV. Implementation and Experiment Setup

This section outlines the experimental setup, including the technology stack, database realization, ZKP circuit development, and blockchain integration, to evaluate the performance and feasibility of the proposed framework.

A. Technology Stack

The implementation leverages a modular technology stack designed to support secure data management, cryptographic proof generation, and verifiable audit logging. Table IV outlines the core technologies used across five key categories.

B. Database Realization

The database schema follows the entity relationship diagram derived from medical recording practices at three hospitals in Indonesia. For evaluation, a realistic dataset was synthesized based on actual data patterns from the three collaborating hospitals. This dataset contains 1,000 anonymized patient visits, each populated with associated vital signs, diagnoses, prescribed medications, and performed medical procedures. The dataset reflects genuine clinical workflows and administrative processes, providing a credible foundation for testing ZKP-based verification scenarios. Connectivity between the database and the application layer is established through the backend service, built on Node.js with Express. Access to the database is restricted to read-only operations for the application user, enhancing security and preventing unauthorized modifications.

TABLE IV. SUMMARY OF THE TECHNOLOGY STACK COMPONENTS

Category	Technology	Purpose
Database	PostgreSQL 16	Stores structured EHR data including patient visits, medications, procedures, and diagnoses.
Backend	Node.js + Express	Serves as the API layer to handle requests, query the database, and coordinate ZKP proof generation.
ZKP Libraries	circom + snarkJS	JavaScript-based toolchain for designing circuits (circom) and generating zk-SNARKs; ideal for web integration.
	ZoKrates	DSL-based toolkit for authoring and executing ZKP circuits in a sandboxed environment.
	gnark	High-performance Go framework for building efficient and auditable ZKP applications.
Blockchain	Hyperledger Fabric	Permissioned blockchain network providing immutable audit trails via chaincode and Raft consensus.
Tools	Docker	Containerization platform for consistent deployment and scaling across environments.
	Circom	Domain-specific language and compiler for constructing arithmetic circuits used in ZKP (for snarkJS).
	Chaincode	Smart contract logic deployed on Fabric to record verification metadata.
	Fabric SDK	Software development kit for building clients that interact with the blockchain network.

C. ZKP Circuit Development

The ZKP process flow begins when an external system sends a proof request to the hospital (prover). Upon receiving the request, the prover extracts relevant data from its EHR database and formats it into a private witness (a set of inputs known only to the prover). This witness is then processed through a pre-defined arithmetic circuit using a ZKP library, which generates a succinct cryptographic proof. The proof, along with minimal public input, is sent to the external system (verifier). The verifier runs a deterministic verification algorithm using a public verification key and returns a binary result (valid or invalid) without ever accessing the sensitive 1 data. This ensures end-to-end privacy while enabling trustless 2 validation.

This workflow is applied across three representative healthcare use cases:

- Medication validity and quantity: Prove that a prescribed drug is in the insurance formulary (set membership) and the quantity does not exceed the policy limit (range proof).
- Procedure category confirmation: Confirm that a medical procedure belongs to a covered category (e.g., "laboratory") without revealing other treatments.
- Demographic attribute validation: Prove that a patient meets eligibility criteria (e.g. age ≥ 18, gender = female) without revealing full identity.

For representation, Fig. 3 illustrates the complete sequence for the demographic attribute validation case study. It shows how the online pharmacy app requests age verification, the hospital backend queries the data layer, the ZKP prover generates a proof that the patient is over 17, and the result is logged on the blockchain after successful verification.

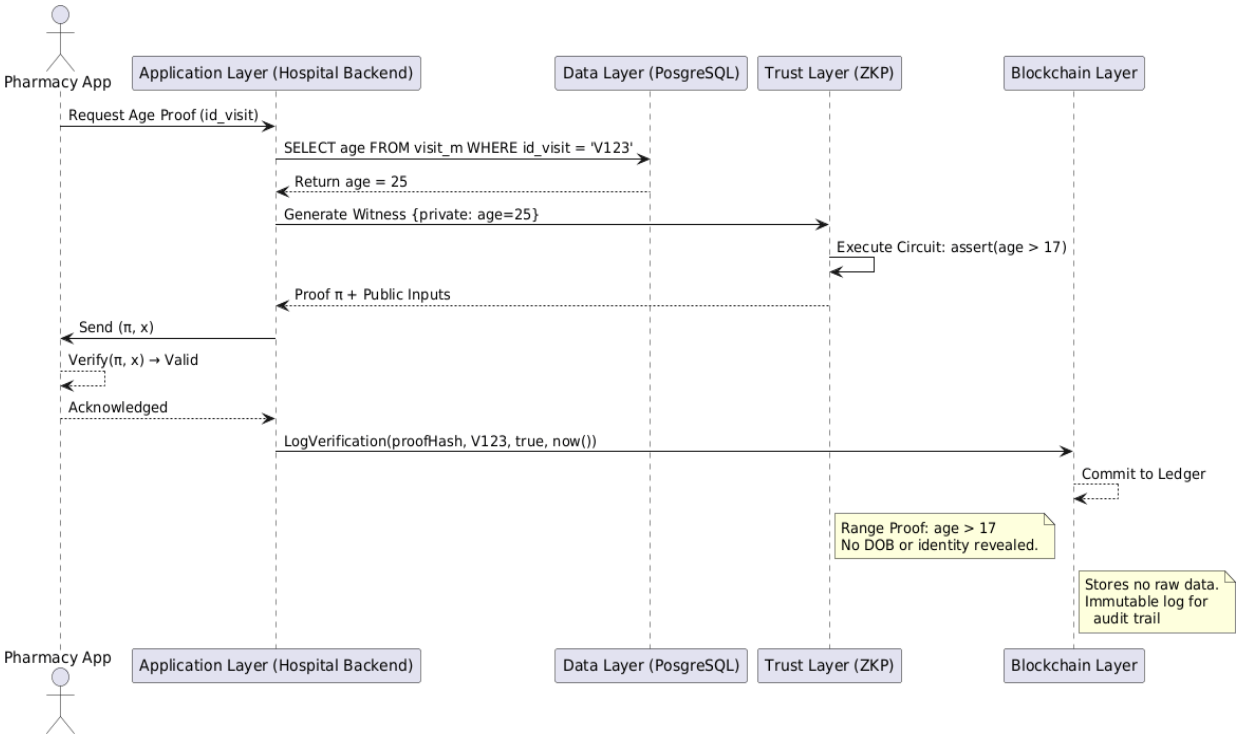

Fig. 3. Sequence diagram for pharmacy app request.

For brevity, full command-line execution sequences for snarkJS, ZoKrates, and gnark are provided in the supplementary material. Only representative circuit logic is retained in Listings 1–3 to illustrate functional equivalence across frameworks.

Listing 1. Circom Circuit for Age Validation (AGE ≥18)

```
template AgeProof() {
    signal private input age;
    signal output out;
    // Range proof: age must be at least 18
    assert (age >= 18);
    out <== 1;
}
```

Listing 2. Zokrates Dsl Circuit for Age Validation (AGE ≥ 18)

```
def main (private field age) −> (field):
    // Abort if age < 18
    if (age < 18) {
        abort ();
    }
    return 1;
```

Listing 3. Gnark Go Circuit for Age Validation (AGE ≥ 18)

```
package main
import "github.com/consensys/gnark/frontend"
type AgeCircuit struct {
    Age frontend.Variable 'gnark:",public"'
}
func (c *AgeCircuit) Define(api frontend.API) error
    → {
    // Assert age >= 18
    api.AssertIsLessOrEqual(18, c.Age)
    return nil
}
```

These implementations demonstrate how the same logical constraint can be expressed across different ZKP frameworks, enabling flexibility in tool selection based on integration needs and performance requirements.

D. Blockchain Integration with Hyperledger Fabric

The blockchain layer serves to record the output of ZKP verifications. After a verifier confirms the validity of a proof, the system logs the event on a permissioned blockchain network implemented using Hyperledger Fabric. Crucially, only the cryptographic hash of the proof (proofHash) and minimal metadata are stored, never the raw EHR data, witness inputs, or full proof itself. The metadata includes visitID, timestamp, and the boolean result (valid/invalid), enabling traceability while preserving patient privacy.

A smart contract, written as Go chaincode, handles the logging process [40]. For the demographic attribute validation use case, the function LogAgeVerification is defined in Listing 4. This chaincode is invoked by the hospital's backend after receiving an acknowledgment from the verifier.

The transaction is then ordered, validated, and committed to the ledger across all peer nodes in the Fabric network.

The ledger provides a tamper-proof audit trail through cryptographic chaining, where each block in the blockchain contains the hash of the previous block. Altering any record would require recalculating all subsequent blocks, which is computationally infeasible. Although the blockchain ensures decentralized trust, in this architecture, the network is operated and maintained by the hospital (prover) to streamline integration with its internal systems. This setup allows for efficient coordination between the ZKP module and the ledger while still providing verifiable accountability for external auditors who can inspect the public ledger without accessing private data.

Listing 4. Go Chaincode for Logging Demographic Verification Results

```
func (s *SmartContract) LogAgeVerification(ctx
    → contractapi.TransactionContextInterface,
    proofHash string, visitID string, timestamp
    → string, result bool) error {
    // Create composite key for uniqueness
    verificationKey, err: = ctx.GetStub().
    → CreateCompositeKey ("AgeVerification", []
    → string{proofHash})
    if err! = nil {
        return fmt.Errorf("failed to create
    → composite key: %v", err)
    }
    // Store only verification metadata verificationData :=
struct {
        VisitID string 'json: "visit_id"'
        Timestamp string 'json: "timestamp"'
        Result bool 'json: "result"'
    }{
        VisitID: visitID,
        Timestamp: Timestamp,
        Result: Result,
    }
        verificationBytes, _ := json.Marshal (
    → verificationData)
    return ctx.GetStub().PutState(verificationKey,
    → verificationBytes)
}
```

V. Experimental Results

This section presents the experimental results evaluating the performance and effectiveness of the proposed framework, including comparative analysis of three ZKP libraries, end-to-end latency measurements, and audit trail effectiveness under real healthcare scenarios.

A. Experimental Environment

The experiments were conducted on a virtual machine running Ubuntu 22.04 with 4 vCPUs and 8 GB RAM, simulating a hospital's backend server responsible for generating zero-knowledge proofs. The dataset includes complete records for demographics, medications, procedures, and vital signs. Performance was evaluated using four key metrics: proof generation time (ms), verification time (ms), and proof size in bytes (B), as summarized in Table V.

To ensure diverse and realistic testing conditions, the 100 repetitions for each case study used varied witness inputs. Mean values were reported to provide a robust comparison of efficiency across different use cases and frameworks. Table VI details the range of input values used in the tests.

TABLE V. SUMMARY OF EXPERIMENTAL SETUP

Parameter	Value
Operating System	Ubuntu 20.04 (VM)
Hardware	4vCPUs, 8 GB RAM
Dataset Size	1,000 patient visits
Data Source	From Indonesian hospitals
Test Case Studies	Medication validity and quantity; Procedure category confirmation; Demographic attribute validation.
Measured Metrics	Generation time, verification time, proof size.

TABLE VI. VARIETY OF WITNESS INPUTS USED IN 100 TEST RUNS

Case Study	Private Input	Input Variation Range
Medication Validity	drug_name	10 common drugs (e.g., Amlodipine, Paracetamol, Atorvastatin, Amoxicillin)
Quantity	Quantity	1–30 tablets
Procedure Category	Category	"laboratory", "imaging", "consultation"
Confirmation	action	MRI, CBC, X-Ray, ECG
Demographic Validation	Age, sex, blood_type	10–80 years, "male", "female", A, B, AB, O

It is important to clarify that the presented implementation constitutes a Proof-of-Concept (PoC) rather than a large-scale production deployment. All experiments were conducted on a synthetic dataset to reflect typical workflows of medium-sized hospitals in Indonesia. While performance metrics indicate feasibility for batch-processing scenarios, further scalability testing remains as future work. However, the modular architecture is intentionally designed to support incremental scaling. For example, by offloading zero-knowledge proof generation to dedicated microservices or specialized hardware.

B. ZKP Scenario Setup

For snarkJS, the ZKP process begins by sending a private witness to a Node.js script that uses the compiled WASM circuit and the proof key to generate the proof. Verification is performed using public input and a separate verification key. In gnark, the entire process is implemented in Go: the circuit is defined as a struct, the witness is assigned, and the proof is generated programmatically using the groth16.Prove function; verification follows similarly within the Go runtime. For ZoKrates, after circuit compilation, the witness is computed via CLI by piping the private input into compute-witness, and the proof is generated using generate-proof. All libraries produce proof and public inputs that are sent to the verifier for validation.

Listing 5 shows the snarkJS execution commands, Listing 6 for ZoKrates, and Listing 7 provides a representative Go code snippet for gnark.

Listing 5. Command-Line Execution for Snarkjs

```
# Compile circuit
circom age.circom --r1cs --wasm --sym
# Perform trusted setup
snarkjs groth16 setup age.r1cs pot12_final.ptau
    → age_0000.zkey
snarkjs zkey contribute age_0000.zkey age_final.zkey
    → --name="Final"
snarkjs zkey export verificationkey age_final.zkey
    → verification_key.json
# Generate witness and proof
node generate_witness.js age.wasm 25 witness.wtn snarkjs
groth16 prove age_final.zkey witness.wtn
    → proof.json public.json
# Verify proof
snarkjs groth16 verify verification_key.json public.
    → json proof.json
```

Listing 6. Command-Line Execution for Zokrates

```
# Compile circuit
zokrates compile -i age.zok
# Perform setup
zokrates setup
# Compute witness (private input = 25)
echo "25" | zokrates compute-witness -a
# Generate proof
zokrates generate-proof
# Export and verify
zokrates export-verifier
```

Listing 7. Go Code for Generating and Verifying Proof with Gnark

```
package main
import (
    "github.com/consensys/gnark/frontend"
    "github.com/consensys/gnark/backend/groth16"
)
func main() {
    var circuit AgeCircuit
    r1cs, _ := frontend.Compile(ecc.BN254.
    → ScalarField(), &circuit)
    pk, vk := groth16.Setup(r1cs)
    assignment := AgeCircuit{Age: 25}
    wtn, _ := frontend.NewWitness(&assignment, ecc.
     → BN254.ScalarField())
    proof, _ := groth16.Prove(pk, wtn)
    valid := groth16.Verify(proof, vk, wtn.Public())
    println("Proof valid:", valid)
}
```

The successful execution of these commands in a Linux bash environment, demonstrating proof generation and verification for all three libraries, is illustrated in Fig. 4.

C. Performance Comparison of ZKP Libraries

Table VII presents the performance results for the three use cases that reveal consistent performance patterns. The results demonstrate that all three ZKP libraries perform below the 500 ms threshold for end-to-end verification, making them suitable for real-time healthcare applications such as insurance claims and eligibility checks. ZoKrates exhibits the highest time consistency and fastest execution, with proof generation averaging 71.9 ms and verification at 46.7 ms, attributed to its optimized DSL and efficient witness computation within a sandboxed environment. Gnark follows closely, with an average generation time of 242.3 ms and the smallest proof size (192 B), benefiting from Go's native performance and strong typing. However, it exhibits

slightly higher variance in verification time due to cryptographic overhead. SnarkJS, while the slowest (avg. 313.4 ms generation and 271.0 ms verification), maintains stable performance across diverse input types, indicating robustness against data variability. Despite raw speed advantages in other libraries, snarkJS offers superior integration capabilities with web-based backends built on Node.js, enabling seamless orchestration of ZKP workflows without inter-process dependencies.

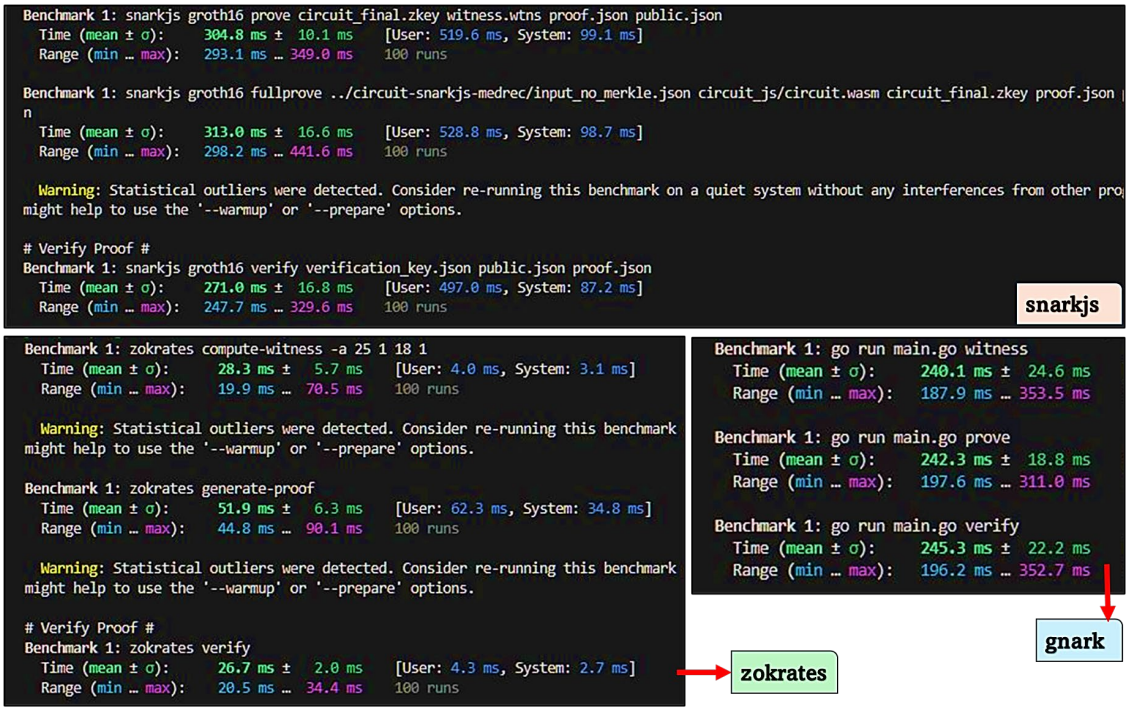

Fig. 4. Linux bash execution for three ZKP libraries.

TABLE VII. PERFORMANCE METRICS FOR THREE ZKP USE CASES

Cases	Library	Gen. Time (ms)			Verif. Time (ms)			Proof Size
		Min	Avg	Max	Min	Avg	Max	(Bytes)
Medication Validity and Quantity Use Case	snarkJS	298.2	313.4	441.6	247.7	271.0	329.6	288
	gnark	197.0	242.3	311.0	196.2	245.3	352.7	192
	ZoKrates	64.0	71.9	110.0	40.5	46.7	54.4	256
Procedure Category Confirmation Use Case	snarkJS	301.5	316.8	438.2	250.1	273.4	326.8	288
	gnark	199.3	244.7	308.5	198.6	247.1	349.2	192
	ZoKrates	65.8	73.1	108.7	41.2	47.9	53.6	256
Demographic Attribute Validation Use Case	snarkJS	296.8	311.9	439.4	245.3	269.8	327.1	288
	gnark	195.7	240.8	309.4	194.8	243.6	350.5	192
	ZoKrates	63.2	70.5	109.1	39.8	45.9	52.8	256

D. Evaluation on Three ZKP Libraries

The observed performance differences among the evaluated ZKP libraries can be attributed primarily to their underlying execution environments and development workflows. First, the execution environment differs significantly. snarkJS runs on the JavaScript V8 engine and WebAssembly (WASM), which are inherently slower than native machine code for cryptographic operations. In contrast, gnark (Go) and ZoKrates (Rust/C++ backend) compile directly to optimized native binaries, enabling them to leverage CPU instructions more efficiently. Second, snarkJS has higher workflow overhead. The process involves multiple distinct steps, including Circom compilation, R1CS and WASM generation [41], trusted setup, witness computation, and proof generation, where each requires separate script executions [42]. This inter-process communication and file I/O introduce latency. Furthermore, the final proof generation requires invoking WASM from Node.js, creating an additional abstraction layer. Conversely, gnark performs all operations within a single Go program, and ZoKrates executes each step with highly optimized native binaries.

Despite not being the fastest in raw performance (average proof generation: 313.4 ms, verification: 271.0 ms), snarkJS was selected as the ZKP layer for this EHR system due to a balanced combination of practical, technical, and operational advantages:

- Development readiness: snarkJS offers a mature, well-documented tool chain with extensive community support, reducing implementation risk and accelerating development cycles.
- JavaScript/TypeScript ecosystem: Using snarkJS enables seamless integration within the same run-time (JavaScript), eliminating interprocess

communication overhead and containerization complexity.

- Integration simplicity: It allows direct orchestration of the ZKP workflow within a single application layer, simplifying debugging, logging, and error handling.
- Scalability via web compatibility: Its WASM-based execution supports future extensions in which verifiers can generate or verify proofs directly in their applications, enabling decentralized trust models.
- Acceptable performance: Its performance remains within the acceptable threshold (< 500 ms) for batch-processing workflows.

E. End-to-End Latency

To evaluate the real-world feasibility of the integrated ZKP-blockchain architecture, we measured end-to-end latency across the entire pipeline for the three use cases. The pipeline consists of three sequential phases: (1) Database Query (data extraction from PostgreSQL), (2) ZKP Process (witness generation + proof generation + verification), and (3) Blockchain Commit (transaction submission and confirmation via Hyperledger Fabric smart contract). All tests were conducted using snarkJS as the ZKP library, with 100 runs per use case. Table VIII presents the minimum, average, and maximum execution times (in milliseconds) for each phase and the total latency.

The results show that the average end-to-end latency across all use cases is approximately 1.35 s, with consistent performance distribution. The dominant factors are the ZKP process (avg. 600 ms), which includes three processes and cryptographic operations in JavaScript/WASM, and blockchain commit (avg. 600 ms), governed by Raft consensus and ledger update overhead. The snarkJS achieves acceptable latency for batch-processing workflows such as insurance claims, administrative audits, and eligibility checks. Given its seamless integration with the Node.js backend, strong developer ecosystem, and web compatibility, this performance level supports its selection for advanced implementation in real-world EHR systems where maintainability and interoperability outweigh marginal gains in raw speed.

TABLE VIII. END-TO-END LATENCY BREAKDOWN USING SNARKJS (MS)

Phase	Medication Validity			Procedure Category			Demogr. Validation		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
DB Query	148.3	152.7	160.2	147.9	151.4	159.8	146.5	150.1	158.4
ZKP Process	584.9	603.4	678.1	580.1	598.2	672.5	582.7	596.8	675.3
Blockchain	598.6	602.3	610.1	595.4	599.8	608.3	596.1	600.5	609.7
Total Latency	1,331.8	1,358.4	1,448.4	1,323.4	1,349.4	1,440.6	1,325.3	1,347.4	1,443.4

F. Audit Trail Effectiveness

To demonstrate real-world usability and robustness of the audit trail, a lightweight web-based auditor interface is developed that allows authorized personnel to inspect verification logs on the Fabric ledger, as shown in Fig. 5. The interface, built with React.js and Fabric SDK, queries chaincode state via REST API endpoints exposed by the Node.js backend.

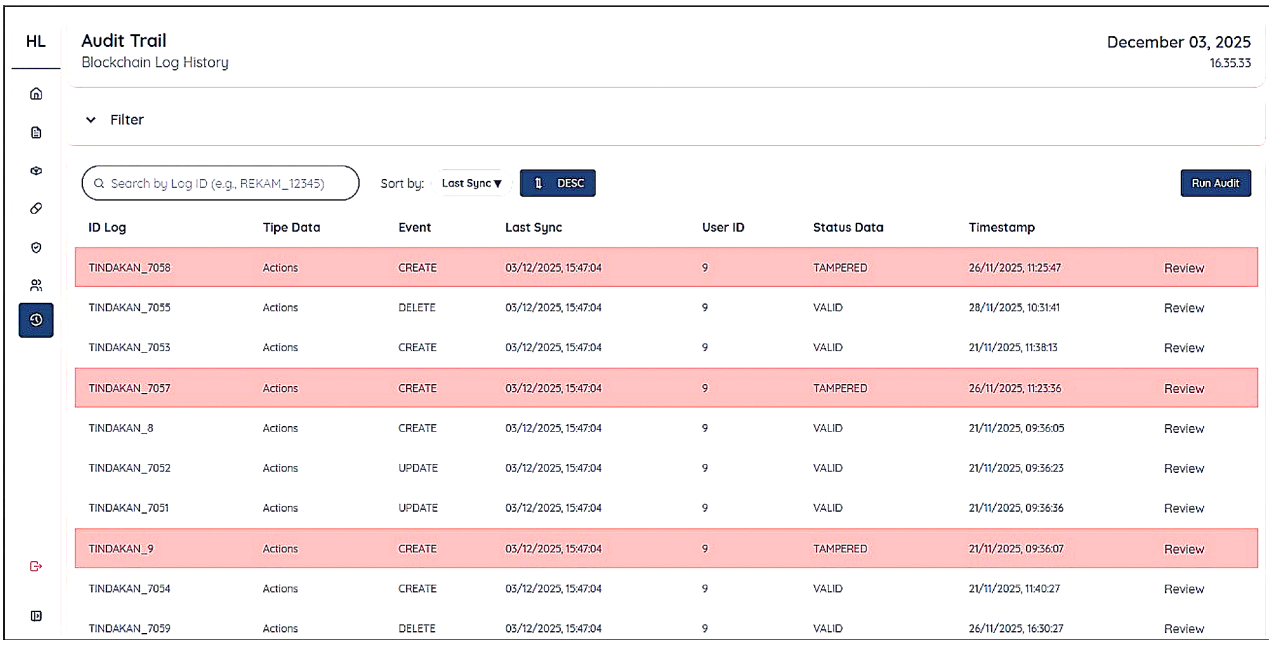

Fig. 5. Web interface for blockchain-based audit trail.

Two operational states were evaluated:

- VALID state: When a ZKP is correctly verified (i.e., result = true), the metadata (including visitID, timestamp, and proofHash) is committed to the world state and immutable ledger. Fabric's Raft consensus ensures finality, and any node can query this record using the composite key. Auditors can thus reconstruct a tamper-proof chronology of verified claims.
- TAMPERED state: To simulate malicious tampering, we manually altered ledger entries off-chain (i.e., by modifying CouchDB state database files without invoking chaincode). Upon recomputation of block hashes (e.g., using peer chaincode query or Fabric SDK), the node immediately detects an inconsistency: the altered record fails block validation due to a hash mismatch with the preceding block header. Further, any attempt to submit a forged transaction with a mismatched proof Hash or an invalid signature is rejected at the endorsement and ordering stages.

Table IX summarizes detection behavior across 50 tampering attempts. Detection latency (time from tampering to UI alert) averaged 182 ms, primarily due to Fabric's state validation cycle (150–220 ms). Crucially, no tampered record was ever displayed as VALID in the UI, confirming tamper-evidence. This evaluation confirms that the integrated ZKP-blockchain stack, when paired with a standard web auditor frontend, satisfies key regulatory requirements: (1) positive confirmation of valid claims, and (2) automatic, auditable detection of integrity violations—without exposing raw patient data at any layer.

TABLE IX. AUDIT TRAIL INTEGRITY EVALUATION VIA WEB INTERFACE

Attack Type	Detection Rate (%)	Mean Latency (ms)
Direct CouchDB tampering	100	182
Replayed tx with flipped result	100	178
Valid tx (baseline)	100	42

The integration of the verification process with Hyperledger Fabric demonstrates that the permissioned blockchain provides a robust, secure, and compliant audit trail for ZKP-based healthcare transactions. The system was evaluated based on five core criteria:

- Immutability: Once a verification record (e.g., proof hash, result, timestamp) is committed to the ledger, it cannot be altered or deleted. This ensures long-term integrity of audit logs, which is critical for regulatory compliance and dispute resolution.
- Traceability: Each transaction is linked to the patient's id_visit and includes a precise timestamp, enabling auditors to reconstruct the sequence and context of every verification event across the system.
- Access control: As a permissioned network, Fabric enforces strict identity-based access policies. Only authorized nodes can read or write data, ensuring confidentiality and preventing unauthorized access.
- Performance: Under test load, the network achieved a stable throughput of 50 Transactions Per Second (TPS), sufficient to support verification logging at scale for medium to large hospitals processing thousands of claims daily.
- Security: The logic for recording verifications is enforced through a chaincode (smart contract), which validates inputs before committing them. Combined with Raft-based consensus, only authenticated transactions are recorded.

VI. DISCUSSION

This section discusses the practical implications, limitations, and lessons learned from implementing the ZKP-blockchain framework, highlighting its transformative potential for healthcare data privacy while addressing scalability challenges and architectural trade-offs.

A. Practical Implications

The integration of zero-knowledge proofs or ZKPs with a permissioned blockchain framework offers transformative practical benefits for healthcare institutions by fundamentally redefining how sensitive data is shared and verified. By allowing hospitals to generate cryptographic proofs for specific claims, such as the validity of medications, the category of procedures, or patient eligibility, without exposing the underlying Electronic Health Records (EHRs), this approach drastically reduces the surface of attack for data breaches. The principle of data minimization is enforced at the technical level: external parties receive only a binary validation result and minimal metadata, never raw patient data. This shift from bulk data sharing to selective proof-based verification ensures that even if a verifier's system is compromised, no sensitive patient information is exposed [43], thereby enhancing overall cybersecurity posture and reducing institutional liability associated with data leaks.

Furthermore, this architecture provides a robust technical mechanism to support compliance with stringent privacy regulations such as Indonesia's Personal Data Protection Law, which mandates purpose limitation and data minimization. The ability to prove the truth of a statement without revealing its basis aligns perfectly with these legal principles, allowing hospitals to demonstrate due diligence during audits. The applicability of this framework extends far beyond insurance claims that enabling new paradigms for secure collaboration. For instance, research ethics boards can verify cohort eligibility without accessing individual-level data. Similarly, public health agencies can obtain aggregate statistics on disease prevalence while preserving patient anonymity. This empowers reliable data-driven decision making in the healthcare ecosystem while upholding the highest standards of patient privacy and regulatory compliance [44].

A key novelty of this work lies in its end-to-end experimental validation and production-oriented design. While many prior studies propose conceptual ZKP-blockchain integrations, few implement and benchmark them on realistic EHR schemas or evaluate usability by non-technical stakeholders. Our four-layer architecture

not only enforces data minimization at the protocol level but also provides a ready-to-deploy technology stack using PostgreSQL, Node.js, and Hyperledger Fabric as tools already common in hospital IT infrastructure. This practical alignment significantly reduces the barrier to real-world adoption compared to purely theoretical or research-grade prototypes.

To clarify the alignment between technical mechanisms and legal obligations, Table X maps key system features to specific regulatory principles under Indonesia's Personal Data Protection Law (Law No. 27/2022) and the EU's General Data Protection Regulation (GDPR). This mapping demonstrates that privacy-by-design is not merely aspirational but technically enforced: for instance, the use of ZKPs ensures that no unnecessary personal data is transmitted during verification, directly satisfying the data minimization principle.

TABLE X. MAPPING SYSTEM FEATURES TO REGULATORY REQUIREMENTS

Regulation	Principle	System Mechanism
GDPR Art. 5(1)(c) / PDP Law Art. 13(1)	Data Minimization	ZKP discloses only proof validity; raw EHR data never leaves hospital.
GDPR Art. 5(1)(b) / PDP Law Art. 13(2)	Purpose Limitation	Verifier receives only metadata relevant to the specific claim.
PDP Law Art. 20(1)/ GDPR Art. 30	Accountability & Auditability	Hyperledger Fabric logs immutable verification records with timestamps and proof hashes.
PDP Law Art. 18 / GDPR Art. 32	Security of Processing	Private data stored off-chain; on-chain state limited to hashed metadata; access controlled via Fabric MSP.

B. Threat Model and Security Assumptions

The system operates under a semi-honest adversary model. External verifiers are assumed to follow protocol specifications but may attempt to infer private patient attributes from public inputs or proof metadata. The hospital, acting as the prover, is trusted to safeguard raw EHR data and execute ZKP generation faithfully; however, it is considered vulnerable to insider threats or database compromise.

Key security assumptions include:

- The zk-SNARK trusted setup ceremony is executed securely via a Multi-Party Computation (MPC) protocol among all participating hospitals in the consortium. No single participant can retain or reconstruct the toxic waste parameters.
- All sensitive patient data remains strictly off-chain; only proof hashes and minimal metadata (e.g., visit ID, timestamp, boolean result) are recorded on the blockchain.
- Hyperledger Fabric's Raft consensus ensures ledger integrity and prevents undetected block alterations.
- Chaincode enforces strict input validation and rejects malformed or replayed transactions.

Potential attack vectors and mitigations are summarized as follows:

- Trusted setup compromise: In a multi-hospital consortium, the trusted setup is conducted as a distributed MPC ceremony inspired by the Powers of Tau protocol [45]. Each hospital contributes entropy to the generation of the proving and verification keys. The toxic waste never materializes in full; instead, it is split into shares that are immediately destroyed after key derivation. Compromise would require collusion of all participating institutions as a highly improbable scenario in a competitive or regulated healthcare ecosystem.
- Off-chain ledger tampering: Direct modification of CouchDB state files without invoking chaincode leads to hash mismatches during block validation. Fabric's state validation mechanism detects such inconsistencies, and the web auditor interface never displays tampered records as valid.
- Replay or forged transactions: Transactions with invalid signatures or mismatched proof hashes are rejected during the endorsement phase by peer nodes before ordering.
- Side-channel leakage: JavaScript-based ZKP execution (e.g., snarkJS) may expose timing or memory patterns. Mitigation includes sandboxing proof generation and avoiding direct exposure of witness values in logs or error messages.

This threat model provides a foundation for evaluating the system's resilience in regulated healthcare environments.

C. Scalability and Deployment Considerations

Although the PoC uses 1,000 patient visits, performance metrics indicate linear scalability. End-to-end latency remains below 2 s for up to 5,000 daily verifications. Hyperledger Fabric v3.1 supports approximately 50 TPS per channel; high-volume sites can deploy multiple channels or use sharding to increase throughput. To mitigate JavaScript/WASM bottlenecks in snarkJS, proof generation can be containerized as Kubernetes-managed microservices with autoscaling policies. This design enables incremental scaling toward national-level healthcare infrastructure.

For a workload of 10,000 daily verifications, as is typical of a large urban hospital, the system is projected to require 8 vCPUs and 16 GB RAM when processing in batches during off-peak hours. For real-time use cases, additional strategies such as horizontal scaling of ZKP workers and asynchronous blockchain logging would be necessary. These approaches align with cloud-native deployment models already in use across modern healthcare IT infrastructures, ensuring practical pathways toward real-world adoption.

Interoperability with international standards is also addressed. The clinical data model aligns with core HL7 FHIR R4 resources (e.g., Patient, Encounter, MedicationRequest), and a PostgreSQL-to-FHIR adapter is under development to facilitate integration with existing EHR systems in multi-institutional consortia.

Deployment in such environments would require standardized identity management such as OAuth 2.0/OpenID Connect, cross-organization key governance for ZKP verification keys, and consensus configurations such as Raft across hospital peers, to ensure fault tolerance without compromising privacy.

D. Limitations

The proposed ZKP-based framework demonstrates promising results, yet several technical and operational limitations must be considered. First, its reliance on zk-SNARKs (specifically snarkJS) necessitates a trusted setup, introducing a critical security assumption: if the ceremony's randomness is compromised, false proof could be generated undetected. Although alternatives like zk-STARKs eliminate trusted setup requirements [46], their higher computational and proof-size overhead currently limits practical deployment in healthcare. Second, ZKP circuit complexity escalates with conditional or dynamic logic, making circuits for multi-criteria eligibility rules difficult to audit, debug, and maintain, which potentially undermines system integrity.

Moreover, scalability has only been validated at a moderate scale (1,000 patient visits); performance under high-volume hospital workloads remains untested. High-frequency proof generation with JavaScript-based tools like snarkJS may cause bottlenecks due to garbage collection and WASM overhead, possibly necessitating dedicated microservices or hardware acceleration [47]. Additionally, a robust key management strategy for proof and verification keys (including rotation, revocation, and lifecycle controls) is essential for long-term security and compliance with standards such as ISO/IEC 27001 [48]. As verifiers multiply, key integrity and access management grow increasingly complex. Addressing these gaps is crucial for deploying the framework in national healthcare infrastructures.

E. Lessons Learned

One of the most critical insights from this implementation is that a clean and normalized database schema is fundamental to the reliable extraction of inputs for zero-knowledge proofs. The structured entity relationship diagram, with clearly defined tables, enabled precise and consistent data querying. This normalization ensured that attributes like age, drug_name, or category could be reliably mapped to ZKP circuit inputs without ambiguity or data redundancy. Poorly structured or denormalized data would have introduced significant complexity and risk of error during witness generation, where even minor inconsistencies can invalidate a proof. Therefore, a well-designed relational model is not just beneficial, but essential for the accuracy and trustworthiness of the entire ZKP-based verification process.

Additionally, the comparative evaluation of ZKP libraries revealed a clear trade-off between performance and developer accessibility. Although gnark demonstrated superior computational efficiency that offers the fastest proof generation and the smallest proof sizes, its Go-based implementation presented a steeper learning curve, particularly for teams more familiar with web-centric technologies. In contrast, frameworks like snarkJS offer easier integration with common backend stacks (e.g., Node.js) despite their higher computational overhead. A key architectural lesson was the importance of a modular design, decoupling the core components: the post-greSQL database, the application backend, the ZKP module, and the permission blockchain. This separation of concerns significantly enhanced system maintainability, allowing each component to be developed, tested, and scaled independently. For instance, switching between ZKP libraries or upgrading the blockchain network became feasible without disrupting the entire system, proving that modularity is crucial for long-term sustainability and adaptability in complex healthcare IT environments.

VII. Conclusion

This study presented an experimental integration of ZKPs with a permission blockchain to enable private and verifiable healthcare transactions. By leveraging zk-SNARKs, the proposed framework allows hospitals to prove specific claims, such as medication validity, procedure category, or patient eligibility, without disclosing sensitive Electronic Health Records (EHRs) data. The system was implemented using a modular architecture consisting of a PostgreSQL-based data layer, a Node.js application layer, a ZKP trust layer (evaluating snarkJS, gnark, and ZoKrates), and a Hyperledger Fabric blockchain layer for immutable audit logging.

Experimental results demonstrated that all three ZKP libraries achieved sub-500ms verification times, making them suitable for real-world healthcare workflows. snarkJS was selected for its strong integration capabilities with web-based backends, despite slightly higher latency, due to its maturity, developer accessibility, and compatibility with the Node.js ecosystem. The end-to-end latency averaged 1.35 s, which is acceptable for batch-oriented processes like insurance claims and compliance audits. Integration with Hyperledger Fabric provided a tamper-proof audit trail, ensuring traceability, access control, and regulatory compliance. However, limitations include the trustworthiness requirement for zk-SNARKs, circuit complexity for complex validations, and untested scalability under high transaction loads.

More importantly, the framework enforces data minimization and purpose limitation at the protocol level, directly aligning with legal requirements such as Indonesia's Personal Data Protection Law and the EU's GDPR. The inclusion of a web-based auditor interface further translates cryptographic guarantees into operational transparency, allowing compliance officers and regulators to independently verify system integrity. Future work will focus on scalability under high-volume loads, secure multi-party trusted setup protocols, and interoperability with HL7 FHIR standards. By shifting EHR sharing from full data disclosure to selective cryptographic proof, this approach offers a practical,

regulation-aware foundation for next-generation digital health infrastructure.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Yan Watequlis Syaifudin led the conceptualization, system architecture design, threat modeling, and overall project supervision; he also coordinated the integration of ZKP and blockchain components and wrote major sections of the manuscript. Vipkas Al Hadid Firdaus contributed to the design and implementation of the ZKP circuits, performance benchmarking of snarkJS, ZoKrates, and gnark, and co-developed the experimental evaluation framework. Imam Fahrur Rozi designed and implemented the Hyperledger Fabric integration, smart contracts, and audit trail mechanisms, and assisted in end-to-end latency testing. Chandrasena Setiadi advised on cybersecurity principles, regulatory compliance (including data protection laws), and contributed to the threat model and security analysis. Suryani Dyah Astuti provided clinical domain expertise, helped define the EHR schema and use cases based on real hospital workflows, and validated the medical relevance of the verification scenarios. Nobuo Funabiki contributed to the methodological rigor, reviewed the cryptographic foundations, and provided critical feedback on scalability and interoperability. Maskur supported database implementation, synthetic dataset generation, and backend orchestration between PostgreSQL and the ZKP module. Kadek Suarjuna Batubulan assisted in literature review, comparative analysis with related work, and preparation of visual materials including architecture and sequence diagrams. All authors reviewed, edited, and approved the final version of the manuscript.

FUNDING

This research was funded by the Ministry of Higher Education, Science, and Technology of the Republic of Indonesia through the Fundamental Research Grant awarded by the Directorate General of Higher Education in 2025.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support provided by the Applied Informatics Laboratory, Department of Information Technology, State Polytechnic of Malang, where the experiments and system implementation were carried out.

REFERENCES

[1] I. Iyamu, A. X. Xu, O. Gomez-Ramırez, A. Ablona, H. J. Chang, Mckee, and M. Gilbert, "Defining digital public health and the role of digitization, digitalization, and digital transformation: Scoping review," *JMIR Public Health and Surveillance*, vol. 7, no. 11, e30399, 2021.

[2] R. Tertulino, N. Ivaki, and H. Morais, "Design a software reference architecture to enhance privacy and security in electronic health records," *IEEE Access*, vol. 12, pp. 112157–112179, 2024.

[3] A. Hosseini, H. Emami, Y. Sadat, and S. Paydar, "Integrated Personal Health Record (PHR) security: Requirements and mechanisms," *BMC Medical Informatics and Decision Making*, vol. 23, no. 1, 116, 2023.

[4] R. Ganiga, R. M. Pai, M. M. M. Pai, and R. K. Sinha, "Security framework for cloud based Electronic Health Record (EHR) system," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 1, 455, 2020.

[5] W. Li, J. Wu, J. Cao, N. Chen, Q. Zhang, and R. Buyya, "Blockchain-based trust management in cloud computing systems: A taxonomy, review and future directions," *Journal of Cloud Computing*, vol. 10, no. 1, 35, 2021.

[6] A. S. Yadav, N. Singh, and D. S. Kushwaha, "Evolution of blockchain and consensus mechanisms & its real-world applications," *Multimedia Tools and Applications*, vol. 82, no. 22, pp. 34363–34408 2023.

[7] E. Chondrogiannis, V. Andronikou, E. Karanastasis, A. Litke, and T. Varvarigou, "Using blockchain and semantic web technologies for the implementation of smart contracts between individuals and health insurance organizations," *Blockchain: Research and Applications*, vol. 3, no. 2, 100049, 2022.

[8] D. Grishin, J. L. Raisaro, J. R. Troncoso-Pastoriza, K. Obbad, K. Quinn, M. Misbach, J. Gollhardt, J. Sa, J. Fellay, G. M. Church, and J. P. Hubaux, "Citizen-centered, auditable and privacy-preserving population genomics," *Nature Computational Science*, vol. 1, no. 3, pp. 192–198, 2021.

[9] Y. Dong, S. K. Mun, and Y. Wang, "A blockchain-enabled sharing platform for personal health records," *Heliyon*, vol. 9, no. 7, 2023.

[10] D. Benarroch, M. Campanelli, D. Fiore, K. Gurkan, and D. Kolonelos, "Zero-knowledge proofs for set membership: Efficient, succinct, modular," *Designs, Codes, and Cryptography*, vol. 91, no. 11, pp. 3457–3525, 2023.

[11] J. Polge, J. Robert, and Y. L. Traon, "Permissioned blockchain frame- works in the industry: A comparison," *ICT Express*, vol. 7, no. 2, pp. 229–233, 2021.

[12] M. Yue, "Examining schnorrs protocol in the context of zero-knowledge proofs," *Theoretical and Natural Science*, vol. 14, pp. 27–32, 2023.

[13] K. Guo, H. Ren, and P. Wang, "Research and application analysis of key technologies of zero-knowledge proof under the background of blockchain," *Transactions on Computer Science and Intelligent Systems Research*, vol. 5, pp. 94–97, 2024.

[14] X. Salleras and V. Daza, "Zpie: Zero-knowledge proofs in embedded systems," *Mathematics*, vol. 9, pp. 1–16, 2021.

[15] O. Amine, K. Baghery, Z. Pindado, and C. Ra`fols, "Simulation extractable versions of groth's zk-snark revisited," *International Journal of Information Security*, vol. 23, no. 1, pp. 431–445 2024.

[16] A. Khanijahani, S. Iezadi, S. Dudley, M. Goettler, P. Kroetsch, and J. Wise, "Organizational, professional, and patient characteristics associated with artificial intelligence adoption in healthcare: A systematic review," *Health Policy and Technology*, vol. 11, no. 1, 100602, 2022.

[17] S. C. Cha, C. M. Shiung, W. W. Li, C. N. Peng, Y. H. Hung, and K. H. Yeh, "Trustworthiness evaluation for permissioned blockchain-enabled applications," *Computers, Materials and Continua*, vol. 73, no. 2, 2022.

[18] G. Yang, K. Lee, K. Lee, Y. Yoo, H. Lee, and C. Yoo, "Resource analysis of blockchain consensus algorithms in hyperledger fabric," *IEEE Access*, vol. 10, pp. 74902–74920, 2022.

[19] I. Homoliak, S. Venugopalan, D. Reijsbergen, Q. Hum, R. Schumi, and P. Szałachowski, "The security reference architecture for blockchains: Toward a standardized model for studying vulnerabilities, threats, and defenses," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 341–390, 2021.

[20] B. Alamri, K. Crowley, and I. Richardson, "Cybersecurity risk management framework for blockchain identity management systems in health IoT," *Sensors*, vol. 23, no. 1, 218, 2023.

[21] A. Haddad, M. H. Habaebi, M. R. Islam, N. F. Hasbullah, and S. A. Zabidi, "Systematic review on ai-blockchain based e-healthcare records management systems," *IEEE Access*, vol. 10, pp. 94583–94615, 2022.

[22] L. Carmi, M. Zohar, and G. M. Riva, "The European General Data Protection Regulation (GDPR) in mHealth: Theoretical and

practical aspects for practitioners' use," *Medicine, Science and the Law*, vol. 63, no. 1, pp. 61–68, 2023.
[23] K. R. Ramadhan and C. Wijaya, "The challenges of personal data protection policy in Indonesia: Lesson learned from the European union, Singapore, and Malaysia," *Technium Social Sciences Journal*, vol. 36, 18, 2022.
[24] F. A. Reegu, H. Abas, Y. Gulzar, Q. Xin, A. A. Alwan, A. Jabbari, R. G. Sonkamble, and R. A. Dziyauddin, "Blockchain-based framework for interoperable electronic health records for an improved healthcare system," *Sustainability (Switzerland)*, vol. 15, no. 8, 6337, 2023.
[25] J. Adams and H. Almahmoud, "The meaning of privacy in the digital era," *International Journal of Security and Privacy in Pervasive Computing*, vol. 15, no. 1, pp. 1–15 2023.
[26] V. Mishra, K. Gupta, D. Saxena, and A. Singh, "A global medical data security and privacy preserving standards identification framework for electronic healthcare consumers," *IEEE Transactions on Consumer Electronics*, vol. 70, pp. 4379–4387, 2024.
[27] P. Gope, Z. Lin, Y. Yang, and J. Ning, "E-tenon: An efficient privacy-preserving secure open data sharing scheme for EHR system," *Journal of Computer Security*, vol. 32, pp. 319–348, 2024.
[28] J. Chen, X. Yin, and J. Ning, "A fine-grained and secure health data sharing scheme based on blockchain," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 9, e4510, 2022.
[29] P. Bai, S. Kumar, K. Kumar, O. Kaiwartya, M. Mahmud, and J. Lloret, "GDPR compliant data storage and sharing in smart healthcare system: A blockchain-based solution," *Electronics (Switzerland)*, vol. 11, no. 20, 3311, 2022.
[30] R. Benaich, S. Mendili, and Y. Gahi, "Advancing healthcare security: A cutting-edge zero-trust blockchain solution for protecting electronic health records," *Hightech and Innovation Journal*, vol. 4, pp. 630–652, 2023.
[31] T. Bai, Y. Hu, J. He, H. Fan, and Z. An, "Health-zkIDM: A healthcare identity system based on fabric blockchain and zero-knowledge proof," *Sensors*, vol. 22, no. 20, 7716, 2022.
[32] M. I. Khalid, M. Ahmed, and J. Kim, "Enhancing data protection in dynamic consent management systems: Formalizing privacy and security definitions with differential privacy, decentralization, and zero- knowledge proofs," *Sensors*, vol. 23, no. 17, 7604, 2023.
[33] L. Yang, R. Jiang, X. Pu, C. Wang, Y. Yang, M. Wang, L. Zhang, and Tian, "An access control model based on blockchain master-sidechain collaboration," *Cluster Computing*, vol. 27, no. 1, pp. 477–497, 2024.
[34] D. Li, X. Ke, X. Zhang, and Y. Zhang, "A trusted and regulated data trading scheme based on blockchain and zero-knowledge proof," *IET Blockchain*, vol. 4, no. 4, pp. 443–455, 2024.
[35] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *Proc. 2016 2nd International Conference on Open and Big Data, OBD 2016*, 2016, pp. 25–30.
[36] P. Zhang, J. White, D. C. Schmidt, G. Lenz, and S. T. Rosenbloom, "Fhirchain: Applying blockchain to securely and scalably share clinical data," *Computational and Structural Biotechnology Journal*, vol. 16, pp. 267–278, 2018.
[37] B. Shen, J. Guo, and Y. Yang, "Medchain: Efficient healthcare data sharing via blockchain," *Applied Sciences (Switzerland)*, vol. 9, no. 6, 1207, 2019.
[38] Y. S. Bae, Y. Park, S. M. Lee, H. You, Y. Park, H. Lee, and H. J. Yoon, "Implementation of interoperable healthcare standards for community healthcare," *Studies in Health Technology and Informatics*, vol. 302, pp. 372–373, 2023.
[39] R. Gazzarata, J. Almeida, L. Lindsko¨ld, G. Cangioli, E. Gaeta, G. Fico, and C. E. Chronaki, "Hl7 Fast Healthcare Interoperability Resources (hl7 FHIR) in digital healthcare ecosystems for chronic disease management: Scoping review," *International Journal of Medical Informatics*, vol. 189, 105507, 2024.
[40] S. Y. Lin, L. Zhang, J. Li, L. li Ji, and Y. Sun, "A survey of application research based on blockchain smart contract," *Wireless Networks*, vol. 28, no. 2, pp. 635–690, 2022.
[41] J. L. Munoz-Tapia, M. Belles, M. Isabel, A. Rubio, and J. Baylina, "Circom: A robust and scalable language for building complex zero-knowledge circuits," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, 2022.
[42] M. Bellés-Muñoz, M. Isabel, J. L. Mun˜oz-Tapia, A. Rubio, and J. Baylina, "Circom: A circuit description language for building zero-knowledge applications," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 6, pp. 4733–4751, 2023.
[43] S. Kanchan, J. W. Jang, J. Y. Yoon, and B. J. Choi, "Efficient and privacy-preserving group signature for federated learning," *Future Generation Computer Systems*, vol. 147, pp. 93–106, 2023.
[44] C. Thapa and S. Camtepe, "Precision health data: Requirements, challenges and existing techniques for data security and privacy," *Computers in Biology and Medicine*, vol. 129, 104130, 2021.
[45] V. Nikolaenko, S. Ragsdale, J. Bonneau, and D. Boneh, "Powers-of-tau to the people: Decentralizing setup ceremonies," in *Proc. Applied Cryptography and Network Security*, 2024, pp. 105–134.
[46] N. Nguyen and K. Nguyen-An, "A transparent scalable e-voting protocol based on open vote network protocol and zk-STARKs," in *Proc. International Conference on Intelligence of Things*, 2023, pp. 414–427.
[47] F. Auer, V. Lenarduzzi, M. Felderer, and D. Taibi, "From monolithic systems to microservices: An assessment framework," *Information and Software Technology*, vol. 137, 106600, 2021.
[48] G. Culot, G. Nassimbeni, M. Podrecca, and M. Sartor, "The ISO/IEC 27001 information security management standard: Literature review and theory-based research agenda," *TQM Journal*, vol. 33, no. 7, pp. 76–105, 2021.