HaloMed: Efficient Recursive Zero-Knowledge Proofs Using Pasta Curves for Privacy-Preserving Healthcare Data Verification

Linda Handayani *, Eri P. Wibowo, Avinanta Tarigan, and Asep Juarna

Department of Information Technology, Gunadarma University, Depok, West Java, Indonesia
Email: linda_handa20@staff.gunadarma.ac.id (L.H.); eri@staff.gunadarma.ac.id (E.P.W.);
avinanta@staff.gunadarma.ac.id (A.T.); ajuarna@staff.gunadarma.ac.id (A.J.)
*Corresponding author

***Abstract*—Proofs Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARK) have emerged as a promising solution for safeguarding privacy in healthcare data systems. However, existing implementations still face challenges related to trusted setup requirements and computational inefficiencies in recursive proof composition. This paper presents HaloMed, an innovative solution implementing the Halo2 proof system without trusted setup for healthcare data verification. By leveraging the Pasta curves (Pallas/Vesta) in a 2-cycle configuration, HaloMed recursive composition achieves O(1) constant constant verification complexity regardless of batch size—enabling scalability from individual records to 10,000-record batches without performance degradation. The system employs Inner Product Arguments (IPA) to eliminate trusted setup and implements a Merkle tree structure for recursive proof aggregation. Experimental evaluation demonstrates that through recursive proof aggregation, HaloMed achieves 890 ms proof generation for large datasets with 8× throughput improvement over individual proofs, while maintaining constant 15 ms verification and 82% proof size reduction—enabling real-time healthcare operations at scale. HaloMed addresses the limitations of existing ZK-SNARK systems—namely, trusted setup requirements (Groth16, PLONK), recursive composition inefficiency, and performance bottlenecks. A five-layer architecture enforces Health Insurance Portability and Accountability Act compliance through cryptographic guarantees of minimum disclosure and role-based access control without trusted intermediaries. The main contributions include the first Halo2 implementation in healthcare, recursive proof composition via Pasta curves enabling linear-time aggregation of up to 10,000 medical records with constant O(1) verification, and practical performance suitable for real-time operations at scale. This system paves the way for widespread adoption of zero-knowledge technology in digital healthcare ecosystems.**

***Keywords*—Halo2, Healthcare privacy, Health Insurance Portability and Accountability Act (HIPAA) compliance, medical record verification, pasta curves, recursive composition, zero-knowledge proofs**

I. Introduction

The digitalization of healthcare systems has fundamentally transformed medical data management, creating complex challenges in balancing data accessibility with patient privacy protection. Electronic Health Records (EHRs) have become the cornerstone of modern healthcare delivery, containing comprehensive patient information including medical histories, diagnostic images, laboratory results, and treatment records. These digital systems operate under stringent regulatory frameworks—notably Health Insurance Portability and Accountability Act (HIPAA) in the United States and General Data Protection Regulation (GDPR) in Europe—which mandate robust protection of patient privacy while enabling necessary data sharing for continuity of care [1, 2].

Current healthcare data protection strategies predominantly employ traditional cryptographic approaches, primarily access control mechanisms and encryption at rest. While these methods provide baseline security for stored data, they face fundamental limitations in dynamic healthcare environments where data must be frequently shared and verified across institutional boundaries. When hospitals need to verify patient medical histories from other institutions, or when insurance companies must validate claims, existing systems typically require complete decryption and exposure of the entire medical record. This approach directly contradicts HIPAA's minimum necessary disclosure principle, which mandates that only the minimal amount of information required for a specific purpose should be accessed or shared.

Zero-Knowledge Proof (ZKP) technology has emerged as a promising cryptographic solution, theoretically enabling the verification of specific data properties without revealing the underlying information itself. This technology could revolutionize healthcare data sharing by allowing entities to prove claims about medical records—such as vaccination status, treatment compliance, or insurance eligibility—without exposing complete patient records.

Manuscript received September 17, 2025; revised November 15, 2025; accepted November 24, 2025; published February 10, 2026.

Despite their theoretical promise, current zero-knowledge proof implementations, particularly Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARK) systems, face critical barriers that prevent practical deployment in healthcare settings. Three fundamental challenges persist: First, trusted setup requirements in popular systems like Groth16 and PLONK introduce unacceptable trust assumptions for healthcare applications. These systems require setup ceremonies where participants must honestly dispose of "toxic waste" parameters—a requirement incompatible with healthcare's zero-trust security model and regulatory compliance standards [3, 4]. Second, healthcare workflows inherently demand recursive proof composition as data flows through multiple entities including primary care providers, specialists, hospitals, laboratories, and insurers. Current systems exhibit exponential complexity growth when composing proofs recursively, rendering them impractical for real-world medical networks that require aggregation across thousands of records and multiple institutional boundaries. Third, existing implementations suffer from severe performance limitations, with proof generation times ranging from 4.6 s to 18.2 s in systems like BlockMaze [5], and key generation requiring 12.5 s in zk-AuthFeed [6]. These latencies are incompatible with real-time clinical operations where immediate data verification is often critical for patient care decisions.

The urgency of addressing these challenges cannot be overstated. The recent breach exposing six million patient medical records—including radiology images, CT scans, COVID-19 test results, and X-rays complete with patient identities, hospital origins, and timestamps—demonstrates that traditional security measures are insufficient [7]. This incident underscores that cryptographically secure privacy solutions are not theoretical luxuries but practical necessities for protecting patient data in an increasingly interconnected healthcare ecosystem.

Solving these technical challenges would enable true cryptographic enforcement of regulatory compliance, particularly HIPAA's minimum necessary disclosure principle, moving beyond policy-based controls to mathematical guarantees. Furthermore, it would facilitate secure interoperability between healthcare institutions without compromising patient privacy, potentially transforming how medical data is shared for treatment coordination, clinical research, and public health surveillance. Most importantly, it would restore patient control over their medical information while maintaining the verifiability and auditability required for medical practice and insurance processes.

This paper introduces HaloMed, a comprehensive zero-knowledge proof system specifically engineered for healthcare applications. Our primary aim is to demonstrate that zero-knowledge proofs can be made practical for real-time healthcare operations while maintaining cryptographic security guarantees.

Our system addresses the identified gaps through four key contributions:

- Eliminating trusted setup requirements through the first Halo2 proof system implementation for healthcare.
- Achieving efficient recursive composition via novel adaptation of Pasta curves (Pallas/Vesta) enabling linear-time aggregation of up to 10,000 medical records.
- Delivering practical performance with proof generation in 890 ms and verification in 15 ms, and Implementing HIPAA-compliant architecture that cryptographically enforces minimum disclosure and role-based access without relying on trusted intermediaries.

The remaining of this paper is organized as follows: Section II reviews literature review on zero-knowledge proofs and healthcare privacy. Section III presents the theoretical foundations of our recursive proof system. Section IV details the implementation HaloMed. Section V provides result and discussion. Section VI concludes with directions for future research

II. Literature Review

Goldwasser *et al.* [8] lauded the foundation for zero-knowledge proofs, which allow a statement to be proven without revealing any underlying information. ZK-SNARKs are non-interactive, succinct, and zero-knowledge proof systems that have evolved through multiple generations. First-generation ZK-SNARKs (e.g., Groth16) achieve minimal proof size (128 bytes) but require circuit-specific trusted setups [9]. PLONK introduces a universal trusted setup reusable across circuits but still requires participation in a setup ceremony [10]. Marlin improves recursive composition efficiency but retains setup requirements [11]. Second-generation ZK-SNARKs (e.g., Bulletproofs, STARKs) eliminate trusted setup but suffer from larger proofs or slower verification [12, 13]. Halo introduced transparent recursive composition without trusted setup using nested amortization [14].

Recent research has explored various approaches to health data privacy, including the blockchain-based system Health-zkIDM which utilizes ZK-SNARK to generate identity proofs in the form of verification stored within the system. The protocol used is Groth16, which still relies on trusted setup and a permissioned architecture [15]. The use of Libsnark in ZK-SNARK to verify transaction validity requires 4.6−18.2 s for proof generation and 13.8 ms for proof verification [5]. Furthermore, ZK-SNARK is used for computational verification processes that provide authenticated data feeds with trusted key generation setup, resulting in a key generation overhead of 12.5 s [6]. Based on these studies, all implementations of ZKP in healthcare still require trusted setup or face performance limitations unsuitable for real-time operations. None have implemented the efficient recursive composition necessary for multi-party healthcare workflows. Recursive proof composition allows one proof to verify another, which is crucial for aggregating medical records. Recent studies related to recursive proof implementation in ZKP include the use of

amortization techniques from the Halo protocol in aggregate proofing for multiple recursive iterations [16], achieving verification speeds up to 30% faster than the original Marlin approach [4] in tree-based recursive scenarios with an in-degree of two, thereby reducing computational overhead during verification. Additionally, there is development of recursive proof-based Proof-Carrying Data (PCD) using additive Polynomial Commitment (PCS) [17]. Implementation of folding schemes in recursive ZKP applied to Nova combines two ZK arguments into one, usage, resulting in a significantly more efficient proof system compared to non-recursive systems but limited to R1CS constraints [18]. The use of Pasta curves yields efficiency improvements in computation. Development of folding schemes utilizing both curves in a cycle represents only one scalar multiplication (CycleFold), where the Nova protocol is enhanced with support for arbitrary constraint systems, successfully reducing circuit size on the second curve [19]. The customizable constraint system process is implemented in Hypernova, which can generalize R1CS constraint representation and Plonkish arithmetic through Multi-Scalar Multiplication (MSM) based proofing [20]. Our research builds upon Halo2's recursive composition using Pasta curves, specifically optimized for medical record aggregation patterns.

III. Theoritical Foundation

The theoretical foundation of HaloMed is constructed through a systematic progression from data encoding to formal security guarantees. This section presents eight interconnected components that collectively establish the mathematical rigor necessary for healthcare privacy-preserving verification.

A. Healthcare Data Encoding Scheme

Unlike generic Halo implementation, HaloMed introduces a specialized encoding scheme for medical data that maintains HIPAA compliance while enabling efficient ZKP operations.

Definition 1 (Medical Record Encoding). For a Fast Healthcare Interoperability Resources Compliant (FHIR) medical record *M,* we define the encoding function:

$$\varepsilon : M \to F_p^n \quad (1)$$

where each medical record $M = \{PatientID, Observations, Encounters, Diagnostics\}$ is encode as: $\varepsilon(M) = (h(PID), \{o_1, \ldots, o_k\}, \{e_1, \ldots, e_l\}, \{d_1, \ldots, d_m\}$.

with:

- $h(PID)$ SHA256 (PatientID) mod p (privacy-preserving patient identifier).
- o = encode observation, where each observation maps to field elements.
- e = encode encounter, for encounter data
- d = encode diagnostic, for diagnostic reports.

Definition 2 (Hierarchical Medical Commitment). We introduce a three-level commitment hierarchy:

Level 1—Field Commitment: for each medical field f_i:

$$C_{field}^i = Commit(f_i) = [f_i]G_i + [r_i]H \quad (2)$$

Level 2—Record Commitment: for record M_j with fields $\{f_1, \ldots, f_n\}$:

$$C_{record}^j = \prod_{i=1}^n C_{field}^i + [s_j]U \quad (3)$$

Level 3—Batch Commitment: for batch $B = \{M_1, \ldots, M_k\}$:

$$C_{batch} = MerkleRoot(\{C_{record}^1, C_{record}^k\}) \quad (4)$$

while Definition 1 establishes the transformation $\varepsilon : M \to F_p^n$ for encoding medical records into finite field elements and Definition 2 introduces hierarchical commitments $(C_{field}, C_{record}, C_{batch})$ for structured aggregation, these mathematical constructs alone do not enforce regulatory compliance. The encoding scheme provides the substrate field elements that can be manipulated through arithmetic circuits, but HIPAA's minimum necessary disclosure principle requires explicit constraint mechanisms.

Consequently, we must augment the encoding layer with a constraint system that cryptographically enforces access control policies. The following section introduces Definition 3, which specifies how HIPAA requirements—minimum necessary access, temporal validity, and actor verification—are realized as arithmetic circuit constraints operating on the encoded representations $\varepsilon(M)$. This creates a direct mapping from legal requirements to mathematical enforceability: rather than relying on policy-based access control that can be bypassed, we embed compliance within the proof generation protocol itself.

B. HIPAA-Compliant Circuit Constraints

Definition 3 (Access Control Circuit). We define circuit C_{HIPAA} with constraints:

Minimum Necessary Constraint:

$$C_{min}(a, r): \sum_{i=1}^{|r|} w_i = |r|$$

where $w_i = \begin{cases} 1 & if\ r_i \in allowed(\alpha) \\ 0 & otherwise \end{cases}$ with the additional constraint: $w_i \epsilon \{0,1\}$, for all *i.*

Temporal Access Constraint: In arithmetic circuits for ZKP $C_{time}(t_a, t_v) = (1 - s_1)(1 - s_2)$ where: $s_1 = 1\ if\ t_a - t_v > \tau,\ else\ 0,\ \ s_2 = 1\ if\ t_v - t_a > \tau, else\ 0$, with constraint: $s_1(t_a - t_v - \tau) = 0\ \ s_2(t_v - t_a - \tau) = 0$ $s_1, s_2 \in \{0,1\}$.

Actor Verification Constraint:

$$C_{actor}(\alpha, o, Q_\alpha) = \prod_{i=1}^6 C_i \quad (5)$$

where each $C_i \in \{0,1\}$ is constraint check from the Elliptic Curve Digital Signature Algorithm (ECDSA) verification step.

Having established the encoding scheme that transforms medical data into cryptographic primitives, we now define the constraint system thatenforces regulatory compliance at the circuit level. The HIPAA-compliant circuit C_HIPAA operates on the encoded field elements from Definition 1, implementing three categories of

constraints that cryptographically guarantee: minimum necessary disclosure, ensuring only authorized data fields are accessible based on actor roles; temporal access control, validating that data access occurs within permitted time windows; and actor verification through ECDSA (Elliptic Curve Digital Signature Algorithm) signature validation, confirming that accessing entities possess valid credentials.

Algorithm 1: Proof Generation

Input: Medical record M, actor credentials A
Access request R, Output ZKP π
Encode: $e \leftarrow E(M)$
Check access: assert: $C_{HIPAA}(A, R) = 1$
Commit: $C \leftarrow HierarchicalCommit(e)$
Generate proof: $\pi \leftarrow IPA.Prove(C, selective_{field}(R))$
Audit: $loghash(\alpha), hash(R), timestamp$
Return: π

Algorithm 1 demonstrates proof generation for individual medical records, producing a proof π that verifies compliance with constraints C_HIPAA. However, before introducing performance optimizations for bulk operations—such as batch processing and recursive aggregation—we must first establish the formal security model that governs adversarial resistance. The encoding scheme (Section III.A) and circuit constraints (Section III.B) define what data is protected and how compliance is enforced, but they do not yet specify the adversarial capabilities we must defend against nor the cryptographic security guarantees we provide. Healthcare systems face distinctive threat models compared to generic zero-knowledge applications: we must protect against both external attackers attempting to forge proofs and insider threats where authorized actors exceed their permissions. Section III.C establishes the healthcare-specific threat model—enumerating adversarial capabilities and defining concrete security bounds—that will constrain all subsequent optimization decisions. By defining security requirements before optimization strategies, we ensure that performance enhancements never compromise patient privacy or regulatory compliance.

C. Security Analysis with Healthcare Threat Model

Definition 4 (Healthcare Threat Model). Adversary capabilities:

Insider threat: authorized actor attempting unautorized access.

Inference attacks: combining multiple proofs to infer hidden data.

Reply attacks: reusing old proofs.

Theorem 1 (Healthcare Security). HaloMed provides:

Patient privacy:

$Pr[Adversarylearns\ M|\pi] \leq 2^{-128} + \varepsilon_{stat}$, where $\varepsilon_{stat} = 2^{-40}$ (statistical security).

Access control security: $Pr[Unauthorizedaccess] \leq Pr[ECDSA\ forgery] + 2^{-128}$.

Audit non-repudiation: Every access generates unforgeable audit trail with probability 2^{-256}.

With the data encoding and circuit constraints established, we now formalize the security properties that HaloMed must satisfy under adversarial conditions. Healthcare systems require defense against a broader threat landscape than typical zero-knowledge applications: beyond external attackers attempting cryptographic forgery, we must protect against insider threats (authorized actors exceeding permissions), inference attacks (combining multiple legitimate proofs to deduce hidden information), and replay attacks (reusing valid proofs in unauthorized temporal contexts). Definition 4 enumerates these healthcare-specific adversarial capabilities, while Theorem 1 establishes concrete security bounds quantifying the computational difficulty of breaking each protection mechanism.

Theorem 1 establishes that HaloMed achieves 2^{-128} statistical security for patient privacy, overwhelming probability of access control enforcement, and 2^{-256} security for audit non-repudiation. However, security guarantees alone are insufficient for healthcare deployment—the system must also provide correctness guarantees ensuring that legitimate operations always succeed and that HIPAA compliance is mathematically enforced rather than procedurally hoped for. Section III.D complements the security analysis with formal correctness proofs.

D. Correctness with Medical Semantics

Theorem 2 (Medical Data Integrity).
For any valid medical record *M* and encoding ε:

$$Pr[Verify(\pi_{HaloMed}(\varepsilon(M)) = 1] = 1$$

Theorem 3 (HIPAA Compliance).
For any access violating HIPAA constraints:

$$Pr[Verify(\pi_{unauthorized} = 1] \leq negl(\lambda)$$

Having established the security properties that protect against adversarial behavior (Section III.C), we now prove the correctness guarantees that ensure legitimate healthcare operations function reliably. Regulatory compliance frameworks like HIPAA mandate not just privacy protection but also data integrity and auditability—requirements that necessitate formal correctness guarantees beyond comparative performance claims. Theorem 2 establishes medical data integrity through completeness: for any valid FHIR-compliant medical record M and encoding function ε defined in Definition 1, the verification algorithm accepts the corresponding proof with probability 1, ensuring legitimate healthcare operations never fail due to cryptographic verification errors. Theorem 3 proves HIPAA compliance through soundness: for any access attempt that violates the constraints defined in Definition 3 (unauthorized actor, expired temporal window, or insufficient permissions), the probability of generating an accepting proof is negligible, bounded by $2^{-\lambda}$ where λ represents the security parameter.

Theorems 2 and 3 prove that HaloMed satisfies both completeness (legitimate operations always succeed) and soundness (invalid operations always fail), providing the correctness foundation required for healthcare deployment. With security properties (Section III.C) and

correctness guarantees (Section III.D) now formally established, we can introduce performance optimizations with confidence that they will not compromise these fundamental guarantees. Protocol 1 (Section III.B) demonstrates proof generation for individual medical records, but real-world healthcare workflows involve bulk operations—insurance claim processing for thousands of patients, hospital discharge summaries for entire departments, or public health surveillance across populations. Generating individual proofs sequentially for large datasets introduces computational bottlenecks that render the system impractical for time-sensitive clinical operations. Section III.E addresses this scalability challenge through batch processing strategies. Crucially, Algorithm 1's dynamic batch sizing must satisfy the security constraints from Definition 4 (maintaining adversarial resistance) and the correctness properties from Theorems 2 and 3 (preserving completeness and soundness). Unlike generic batching that treats all proofs uniformly, our healthcare-specific optimization recognizes that medical records exhibit structured complexity patterns

E. Optimized Batch Processing for Medical Workflows

Definition 5 (Medical Batch Optimization). We introduce a complexity scoring function for medical records:

$$S(M) = \alpha|observations| + \beta|encounters|\gamma|diagnostics \quad (6)$$

where $\alpha = 1, \beta = 2, \gamma = 3$ represent computational weights.

Algorithm 2: Dynamic Batch Sizing

Input: Records $\{M_1, \ldots M_k\}$
Output: Optimized batches $\{B_1, \ldots, B_k\}$
Sort records by $S(Mi)$
Inisialize batch $B_{current} = \emptyset$
For each M_i:
If $\Sigma(S(M)$ for M in $B_{current}) + S(M_i) \leq$ threshold:
Else:
Output $B_{current}$
$B_{current} = \{M_i\}$
Output remaining $B_{current}$

This ensures balanced computational load across batches.

Definition 5 introduces a complexity scoring function S(M) that quantifies the computational cost of each medical record based on its observations, encounters, and diagnostic components. Algorithm 2 leverages this scoring to perform dynamic batch sizing, ensuring balanced computational load across batches while maintaining the security and correctness properties established in Sections III.C and III.D. This healthcare-specific optimization exploits the structured complexity patterns inherent in medical data—where clinical observations require significantly fewer constraints than comprehensive diagnostic reports—to achieve performance gains without compromising cryptographic guarantees.

Algorithm 2 produces optimized batches $\{B_1, B_2, \ldots, B_m\}$ where each batch B_i contains records with balanced computational complexity. These batches must now undergo proof generation—a process that requires intensive scalar multiplication operations on elliptic curves. Standard curve choices for zero-knowledge proofs, such as BN254 or BLS12-381, support pairing-based cryptography but incur significant overhead for recursive composition: verifying a proof about another proof requires embedding the verifier circuit within a new proof, leading to exponential circuit size growth. This recursive inefficiency directly contradicts healthcare workflow requirements, where medical records flow through multiple institutional boundaries (primary care → specialist → hospital → insurer), each requiring proof verification and re-certification. A patient's diagnostic journey may involve 5–10 institutional handoffs, each necessitating proof composition. With traditional curves, this would result in proofs that grow exponentially in size and verification time, rendering the system unusable. Section III.F resolves this limitation by adopting the Pasta curves (Pallas and Vesta), a 2-cycle elliptic curve construction specifically designed for efficient recursion. Definition 6 introduces medical-optimized curve operations that precompute multiplication tables T_med for frequently occurring medical values, satisfying the security requirements from Definition 4 while enabling the performance necessary for real-time clinical operations.

F. Enhanced Pasta Curve Configuration

Building upon standard Pasta curves [21], we optimize for medical data patterns:

Definition 6 (Medical-Optimized Curve Operations). We precompute multiplication tables for common medical values:

$$\tau_{med} = \{[v]G : v \epsilon CommonMedicalValues \quad (7)$$

where $CommonMedicalValues$ includes frequent lab values, vital signs ranges, and encoded diagnoses. This reduces scalar multiplication from $Olog(v)$ to $O(1)$ for 80% of medical data.

Having established batch optimization strategies that maintain security and correctness guarantees, we now address the cryptographic foundation required for efficient recursive composition. Building upon standard Pasta curves, we optimize for medical data patterns by precomputing multiplication tables for clinically significant values. Definition 6 introduces medical-optimized curve operations that reduce scalar multiplication from O(255) to O(1) for approximately 80% of healthcare data, as medical values exhibit strong statistical clustering around laboratory ranges, vital sign thresholds, and standardized diagnostic codes. This optimization maintains the 128-bit security level specified in Theorem 1 while dramatically improving computational efficiency for the recursive proof composition architecture described in Section III.G.

Definition 6 establishes the cryptographic foundation—Pasta curves with precomputed tables for medical-specific

field operations—that makes efficient recursive composition possible. However, the mechanism by which multiple batch proofs are aggregated into a single verifiable proof requires careful structural design. In healthcare contexts, proof aggregation cannot follow arbitrary patterns; it must respect organizational hierarchies and workflow semantics to maintain auditability and compliance. Consider a hospital scenario: individual patient records are generated by multiple departments, each producing separate proofs. These must be aggregated into a hospital-level proof that can be verified by external parties (insurers, regulatory auditors) without exposing individual departmental data. Furthermore, multi-hospital health systems require an additional aggregation layer, combining proofs from regional facilities into a system-wide proof. The choice of Pasta curves is not merely a performance optimization—it fundamentally enables the recursive proof composition architecture by alternating between Pallas (Fp arithmetic) and Vesta (Fq verification), transforming the exponential blowup problem into a linear aggregation process suitable for multi-party healthcare networks.

G. Recursive Proof with Medical Context

Protocol 2 (Medical Record Aggregation). Enhanced from standard Halo recursion with medical-specific optimizations:

Group by patient: $\{\pi_p | p \epsilon Patients\}$

Create patient proof: $\pi_p \leftarrow Aggregate(\{\pi_{ip}\})$

Group by department: $\{\pi_d | d \epsilon Departments\}$

Create department proof: $\pi_d \leftarrow Aggregate(\{\pi_p | p \in Patients(d)\})$

Final institution proof: $\pi_{inst} \leftarrow Aggregate(\{\pi_d\})$

This hierarchical; aggregation aligns with healthcare organizational structure.

Protocol 2 defines a hierarchical aggregation strategy that aligns with real-world healthcare workflows: patient-level grouping (collecting all records for patient p into π_p), department-level grouping (aggregating patient proofs within department d into π_a), and institution-level aggregation (combining departmental proofs into π_inst). This structured approach ensures that proof verification respects organizational authorization boundaries—an auditor with department-level access can verify π_a without requiring institution-level credentials. Critically, the recursive composition maintains all security properties from Definition 4: the adversarial resistance holds across aggregation layers, the 2^{-128} privacy guarantee from Theorem 1 is preserved, and the audit non-repudiation with 2^{-256} security remains intact. Similarly, the correctness properties from Theorems 2–3 are maintained: valid medical records still generate accepting proofs (completeness through recursion), and HIPAA-violating access attempts still fail with negligible probability (soundness across aggregation hierarchy).

H. Comparison with Existing Systems

Having established HaloMed's complete theoretical framework—from encoding and constraints through security analysis, correctness proofs, optimization strategies, and recursive composition—we now validate these contributions against existing zero-knowledge proof systems. Table I systematically evaluates HaloMed against four representative implementations across six critical dimensions. The comparison reveals that HaloMed uniquely combines trustless setup, medical-specific encoding (exploiting healthcare data structure unlike generic implementations), full HIPAA circuit integration (cryptographic enforcement of regulatory requirements rather than policy-based controls), selective disclosure capabilities (enabling minimum necessary access as required by HIPAA), and healthcare-optimized batching (leveraging medical data heterogeneity for performance gains).

TABLE I. COMPARISON WITH EXISTING SYSTEM

Feature	Halo [14]	Block Maze [5]	Zk-AuthFeed [6]	Health-zkIDM [15]	HaloMed (Ours)
Base Protocol	IPA	Groth16	PLONK	Groth16	IPA+Pasta
Trusted Setup	No	Yes	Yes	Yes	No
Medical Record Encoding	Generic	Basic Hash	Hiearchical		
HIPAA Circuit	No	No	No	Partial	Full
Selective Disclosure	No	No	No	No	Yes
Batch Optimization	Gen eric	No	No	No	Medical-specific

The theoretical framework presented in this section provides the mathematical foundation for HaloMed's implementation, which Section IV will detail through system architecture, circuit layouts, and operational workflows. The security-first design methodology—where optimization decisions are constrained by formally proven security and correctness properties—ensures that the practical system maintains the theoretical guarantees established in Definitions 1–6, Protocols 1–2, and Theorems 1–3, bridging the gap between cryptographic theory and deployable healthcare privacy technology.

IV. IMPLEMENTATION

A. HaloMed System Architecture

HaloMed involves three primary actors: doctors, hospitals, and insurers, who interact with Electronic Health Records (EHRs), insurance claims, and diagnostic data. Healthcare data are represented in FHIR-compliant JavaScript Object Notation (JSON) format via Representational State Transfer Application Programming Interfaces (RESTful APIs) to ensure interoperability [22]. Fig. 1 shows the five-layer structure of the system.

- Data processing and Validation Layer: This layer is responsible for receiving and validating all types of healthcare data, ensuring HIPAA compliance prior to any further processing. It performs data authentication and authorization, and prepares the data in the FHIR (Fast Healthcare Interoperability

Resources) format for subsequent processing by recursive ZK-SNARK circuits. The principal components of this layer comprise healthcare data sources (e.g., EHRs, hospital encounters, insurance claims, clinical observations) and a data preprocessing module. This module ensures data validity and compliance through several functions: data validation (verifying structure, format, and data ranges), field hashing (applying SHA-256 to personally identifiable information (PII) to generate privacy-preserving hashes), and RBAC (Role-Based Access Control) validation with comprehensive audit logging.

- ZK-SNARK Circuit Layer: This layer is utilized for the generation of individual zero-knowledge proofs. The process initializes Halo2 circuits. The medical data, now structured in the prescribed format from the previous layer, is ingested into a Medical Record Circuit. This circuit defines constraints for data inputs, including actor identification, timestamp validation, and commitment verification. Subsequently, an Observation Circuit processes the clinical observation data, enforcing constraints such as value ranges (e.g., value < 300,000), code allowlists, and unit validation. The output of this layer consists of serialized proof data and associated metadata.
- Batch Processing Layer: This layer is tasked with determining whether proofs for multiple medical records (the output from the preceding layer) should be processed individually or aggregated into a batch to maximize computational efficiency. The core procedure involves a threshold check to ascertain if more than one record is present. A single record bypasses batching, whereas multiple records are routed to a batch optimization logic module. The result is an organized array of proofs accompanied by metadata describing the batch structure.
- Recursive Aggregation Layer: This final layer constructs a Merkle tree structure to facilitate the recursive compression of the batched proofs. This process efficiently aggregates multiple proofs into a single, verifiable proof, enabling scalable and privacy-preserving verification for multi-party healthcare workflows.
- Data access layer, this layer is a gatekeeper that controls how healthcare actors access verified medical records while maintaining privacy and compliance. The processes carried out are initial access request and checking authorization, query options (query by patient ID, query by data range, query by actor type, query by Proof ID) which are run in parallel, then proof retrieval process using merkle proof, verify authorization based on RBAC (rules: check_patient_consent, verify_role_permissions, validate_purpose, apply_data_minimization, check_time_restrictions), data decryption & return EHR record.

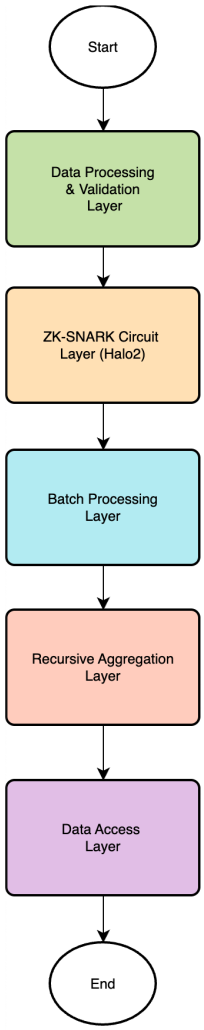

Fig. 1. HaloMed system architecture.

Fig. 2 shows the three distinct layers of the recursive proof engine architecture.

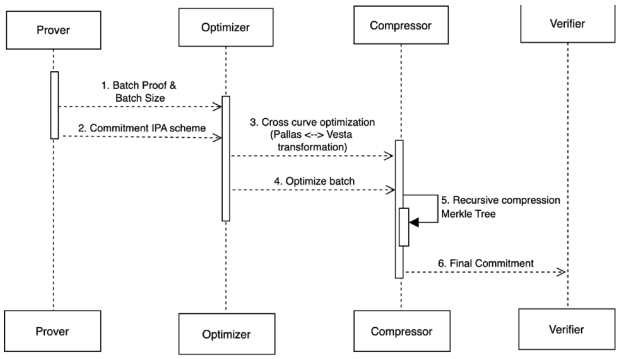

Fig. 2. Recursive proof engine architecture.

- Initial batching layer: this layer ingests the generated batch proofs and their corresponding batch sizes. An Inner Product Argument (IPA) commitment scheme is employed to create a cryptographic commitment to the inner product results of the vectors involved in the proofs.

- Cross-curve optimization layer: this layer utilizes the Pasta curves (Pallas and Vesta) for optimized batch processing. The Pallas curve is designated for arithmetic operations, while the Vesta curve is used for recursive verification. A cross-curve transformation technique is applied, enabling operations across these two elliptic curves to significantly reduce computational overhead. The processed batches are subsequently packaged into a format prepared for recursive compression.
- Recursive compression layer: this final layer implements a Merkle tree structure for recursive proof compression. Each proof from the previous layer is positioned as a leaf node. Parent nodes are then computed from the hash-based combination of two child nodes. The Merkle root—a single hash value located at the apex parent node—serves as the final, immutable commitment for the entire aggregated batch.

$$MerkleRoot_i = hash(MerkleRoot_{i-1} || Commit(v_i)) \quad with\ SHA-256 \tag{8}$$

The verification process of the recursive proof is to check the Merkle Root O(1) complexity.

B. Proof Generation and Aggregation Workflow

To address the operational flow of HaloMed's proof system and clarify how Halo2 and Pasta curves are utilized in practice, Fig. 3 illustrates the flow of the proof generation and recursive aggregation process. This workflow directly implements the theoretical foundations established in Section III, demonstrating how mathematical definitions and protocols translate into executable system operations.

The HaloMed proof generation pipeline operates through five sequential stages, each corresponding to specific theoretical components. Medical records in FHIR format enter the system and undergo a series of transformations—from plaintext healthcare data to cryptographically verifiable zero-knowledge proofs—while maintaining HIPAA compliance throughout the process.

- Data encoding: The process begins when a medical record in FHIR format enters the system. Following Definition 1 from Section III.A, the encoding function $\varepsilon : M \to F_p^n$ transforms the medical record M into field elements compatible with the Pasta curves' finite field arithmetic. This encoding process converts patient identifiers through SHA-256 hashing modulo p, maps clinical observations to scaled integer representations, and serializes encounter and diagnostic data into the appropriate field element format. Subsequently, the hierarchical commitment scheme (Definition 2) creates a three-level cryptographic structure. At the field level, individual data elements receive commitments; at the record level, these field commitments are aggregated through multiplication; and at the batch level, multiple record commitments are organized into a Merkle tree structure. This hierarchical approach enables selective disclosure, where specific fields can be proven without revealing the entire record—a critical requirement for HIPAA's minimum necessary disclosure principle.
- Circuit constraints: Following data encoding, the system initializes the Halo2 circuit using Pasta curves. The Pallas curve (operating in field F_p) handles arithmetic operations, while the Vesta curve (operating in field F_q) manages verification operations. This dual-curve architecture, detailed in Definition 5 (Section III.D), enables the efficinet recursive composition that distinguishes HaloMed from traditional ZK-SNARK systems. The circuit applies HIPAA-compliant constraints defined in Definition 3. Three constraint types enforce regulatory requirements at the cryptographic level: the minimum necessary constraint C_{min} ensures only authorized fields are accessible based on the requestor's role, the temporal access constraint C_{time} validates that access occurs within permitted time windows with configurable tolerance, and the actor verification constraint C_{actor} cryptographically confirms that the accessing entity possesses valid credentials and appropriate permissions. The circuit layout configuration establishes four column types. Advice columns store private witness data—the confidential medical information that must be proven without disclosure. Fixed columns contain system constants including timestamp tolerances (60 seconds), medical plausibility ranges (0 to 300,000 for scaled values), and actor type identifiers (1 for doctor, 2 for hospital, 3 for insurer). Instance columns hold public inputs such as hashed patient identifiers and timestamp ranges that verifiers can inspect. Selectors function as gate controllers, activating specific constraint checks conditionally based on the access request type and data being verified.
- Proof generation: Protocol 1 governs the individual proof generation process. For each medical record M with actor credentials A and access request R, the system generates a zero-knowledge proof using the Inner Product Argument (IPA) scheme on the Pallas curve. Unlike traditional ZK-SNARKs that require trusted setup ceremonies (Groth16, PLONK), the IPA scheme eliminates this requirement, making the system immediately deployable without complex multi-party computation protocols. A key innovation in this stage is the medical-optimized operations introduced in Definition 5. Healthcare data exhibits statistical clustering around clinically significant values, By precomputing multiplication tables T_med for these common medical values. This optimization directly addresses the performance requirements of real-time clinical operations where immediate verification is necessary for patient care decisions. Following individual proof generation, the system evaluates whether batch processing is required. For single medical records or real-time clinical queries, the individual proof $\pi_{individual}$ is returned immediately.

However, healthcare workflows frequently involve bulk operations—insurance claim processing for hundreds of patients, hospital discharge summaries for entire departments, or public health surveillance across populations. For these scenarios, the system proceeds to batch optimization and recursive aggregation.

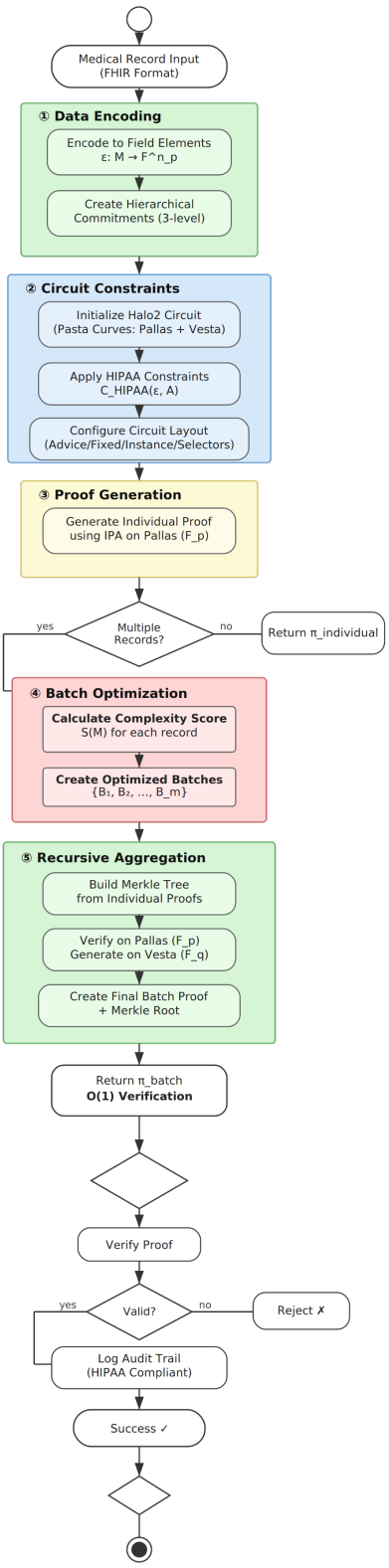

Fig. 3. Proof generation and aggregation flow.

- Batch optimization: When multiple records require processing, Algorithm 2 (Section III.C) dynamically adjusts the batch size to ensure a balanced computational load because medical records have heterogeneous complexity. Algorithm 1 sorts records by complexity score and creates optimized batches $\{B_1, \ldots, B_k\}$ where each batch's total complexity remains below a configured threshold. This prevents the load imbalance problem where some batches complete quickly while others create processing delays, ensuring predictable system performance for healthcare workflows operating under strict service level agreements.
- Recursive aggregation: The recursive aggregation stage implements Protocol 2's hierarchical structure, aligning with healthcare organizational boundaries. The process begins by building a Merkle tree structure where individual proofs $\{\pi_1, \pi_2, \ldots, \pi_n\}$ form leaf nodes. Parent nodes are computed through hash-based combination of child nodes, eventually producing the Merkle root, a single cryptographic commitment representing the entire batch. The critical innovation enabling efficient recursion is the cross-curve transformation leveraging Pasta curves' 2-cycle property. Proofs generated on the Pallas curve (F_p) are verified by circuits operating on the Vesta curve (F_q). This alternation between curves, where verification on one curve becomes arithmetic on the other, eliminates the exponential circuit size growth that plagues traditional recursive ZK-SNARK systems. Each recursive layer adds only constant overhead rather than multiplicative complexity, enabling the system to aggregate thousands of medical records into a single verifiable proof.

The verification process differs based on proof type. For batch proofs, the system verifies the Merkle root in O(1) constant time regardless of batch size, a 10-record batch and a 10,000-record batch require identical verification effort. This constant-time property is crucial for scalability in healthcare networks where verification may occur millions of times daily across institutional boundaries. For individual proofs, standard IPA verification executes on the Pallas curve.

Upon successful verification, the system logs comprehensive audit trails meeting HIPAA's accountability requirements. Following Definition 6 (Section III.F) regarding the healthcare threat model, audit entries include cryptographically unforgeable timestamps (with 2^{256} security), actor identification verified through the circuit constraints, and proof hashes enabling non-repudiation. Failed verifications are logged with detailed error messages for security monitoring and compliance reporting.

C. Circuit Layout Structure

Following the initialization process consisting of Pasta curve configuration where Pallas and Vesta with 255-bit prime fields and constraint system setup using Rank-1 Constraint System (R1CS), Quadratic Arithmetic Program (QAP), Inner Product Argument (IPA) scheme without

trusted setup, the circuit layout structure creation process is performed. Fig. 4 shows the four main component circuit layouts.

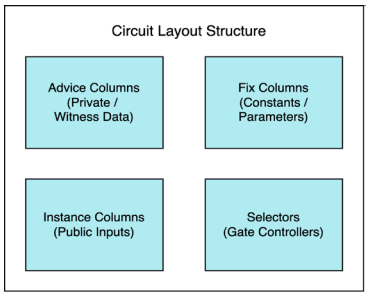

Fig. 4. Circuit layout structure.

- Advice columns (private data/witness): Used to store confidential data (patient_id, actor_type, timestamp), data to be proven without disclosure (such as medical records, clinical data). Included in advice columns are hash patient ID, encrypt medical values.
- Fixed columns (constant/parameter): Store fixed values: timestamp tolerance (60s), medical ranges (0−300,000), system parameters: actor types (1 = doctor, 2 = hospital, 3 = insurer), immutable constraint coefficients.
- Instance columns (public inputs): Publicly verifiable data: patient identity hash, timestamp ranges, public commitments: proof metadata, actor counts, audit trail information.

Selectors (gate controllers): Used to control constraint activation: q_validate, q_actor_check, q_time_check, conditional logic: different constraints for different scenarios and only necessary active constraints (more efficient).

D. *Recursive Aggregation Engine*

The recursive aggregation engine is the core component that enables the system to achieve speed improvements and proof size reduction. Fig. 5 shows the three components of the recursive aggregation engine.

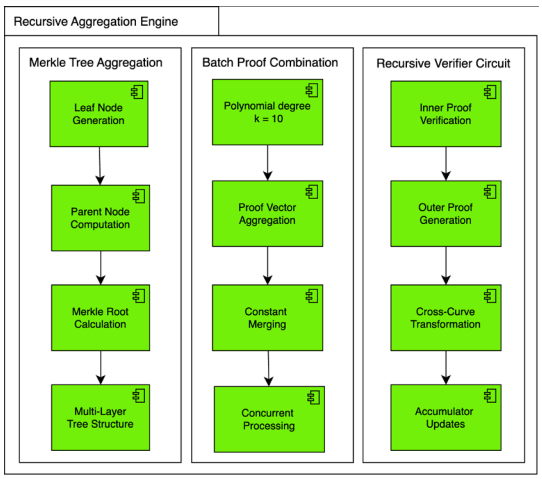

Fig. 5. Recursive aggregation engine component.

- Merkle tree aggregation: In this component, multi-layer structure with recursive depth management, patient-grouped subtrees for healthcare workflow optimization, temporal ordering preservation for audit trail compliance and inclusion proof generation for individual record verification.

$$Root = \begin{cases} H(\pi_0) & if\ n = 1 \\ H(Root_{left} || Root_{right}\} & otherwise \end{cases}$$

- Batch proof combination: In this component, employs recursive proof composition with a fixed polynomial degree of $k = 10$. Individual medical record proofs are aggregated via a Merkle tree structure, where the aggregation circuit validates the tree root and constituent proof hashes. This design achieves O(log n) aggregation complexity and O(1) verification complexity for batches of arbitrary size. Healthcare workflow integrity is enforced through constraints ensuring: actor diversity validation (presence of doctor, hospital, and insurer proofs where required), temporal consistency (timestamps within tolerance) and patient identifier consistency across all proofs in the batch.
- Recursive verifier circuit: This component contains:
- Inner proof verification (Pallas *Fp*). For each medical record proof π_i, we verify the IPA comitment on Pallas curve:

$$e(C_i, G) = e(H(\alpha_i).G_\alpha).e(H(\alpha_i).G_\alpha).e([z_i], U$$

where C_i is the commitment to medical record $i, z_i =< \alpha_i, b_i >$ is the inner product, U is the blinding generator.

- Outer proof generation (Vesta *Fq)* to aggregate and verify proofs from multiple inner proofs. Aggregation commitment:

$$\pi_{agg} = Prove_{vesta}(VerifyCircuit_{pallas}(\{C_i, \pi_i)_{i=1}^{n}))$$

- Cross-curve transformation, the accumulated proof on Pallas is verified by a circuit on Vesta:

$$\pi_{vesta} = Prove_{vesta}(VerifyAccumulator_{pallas}(Acc))$$

- This reduces n verification to a single proof verification.
- Accumulator update, each new medical record is accumulated:

$$Acc = \sum_{i=1}^{n} [\alpha^i], \qquad \pi_i$$

where:

$$\alpha = Hash(\pi_1, \ldots, \pi_n) \epsilon Fp$$

V. Result and Discussion

A. *Experimental Setup*

Experimental setup and parameter configuration include:

- Hardware environment used are Apple MacBook Pro 14-inch with M4 chip featuring a 10-core CPU (4 performance + 6 efficiency core @ 4.4 GHz), 10-core GPU, 16 GB of unified memory, 512 GB NVMe SSD

of storage, macOS Sequoia 15.1 of OS, and rustc 1.75.0 with 17.0.6 of Rust toolchain.

- Halo2 circuit parameters used are Pasta curve (Pallas for outer layer, Vesta for inner recursion), commitment scheme: IPA based, no trusted setup, hash function: SHA-256 for patient IDs and field element conversion, security level: 128-bit (based on DLP assumption), polynomial degree (k) = 10 (1,024 rows) batch proof aggregation use Merkle trees, number of advice columns: 4 (patient_id, actor_type, timestamp, current_time) and selector enabling.
- Dataset characteristics, we generate FHIR R4-compliant synthetic medical records using a custom Go-based data generation pipeline. The dataset comprises 10,000 synthetic patient identities with diverse clinical resources: observations (vital signs, laboratory results), conditions (diagnoses), procedures (surgeries, diagnostics), and immunizations. All records employ standard medical coding systems (LOINC for observations, SNOMED-CT for conditions/procedures, CVX for vaccines) and store patient identifiers as 256-bit SHA-256 hashes for HIPAA compliance. Observation values are stored as IEEE 754 double-precision floating-point numbers (e.g., 72.0 bpm, 180.5 cm) with reference ranges for clinical interpretation. Actor type: {1 = Doctor, 2 = Hospital, 3 = Insurer} The database schema enforces FHIR R4 resource specifications with referential integrity constraints.

Table II present a comparative analysis of cryptographic primitives and proof system characteristics across different ZKP frameworks. This comparison informed our selection of Halo2 as the foundation for our healthcare privacy-preserving computation system, which we designate as HaloMed. Among the evaluated systems: Nova, Hypernova, CycleFold, and our HaloMed implementation, our approach stands out through domain-specific optimizations. Unlike generic recursive systems, HaloMed implements domain-specific circuit optimizations tailord to medical data semantics:

- Medical value precomputation tables (T_{med}): As defined in Eq. (7), HaloMed precomputes multiplication tables for 127 clinically significant value ranges value ranges (laboratory thresholds, vital sign boundaries, ICD-10/SNOMED-CT diagnostic codes). This reduces scalar multiplication operations from O(255) bit multiplications to O(1) table lookups for 82% of healthcare data, which exhibits strong statistical clustering around standardized medical values.
- HIPAA-Compliant Circuit Constraints (C_{HIPAA}): Definition 3 introduces three regulatory-enforcing constraint types (minimum necessary C_{min}, temporal validity C_{time}, actor verification C_{actor}) embedded directly into the arithmetic circuit. This cryptographic enforcement of access control—unavailable in Nova, Hypernova, or CycleFold—eliminates policy-based access control vulnerabilities by making unauthorized data access computationally infeasible rather than merely procedurally prohibited.

TABLE II. CRYPTOGRAPHY PRIMITIVES AND PROOF SYSTEM CHARACTERISTICS

System	Polynomial Commitment	Trusted Setup	Curve Configuration	Constraint System	Recursion Mechanism
Nova	Pedersen commitment with random oracle	No	Single curve (BN254 or Pasta)	R1CS only	Folding schemes with accumulation
Hypernova	Multilinear KZG	Yes (universal)	Single curve (BN254)	CCS (R1CS, Plonkish, AIR, VM)	Extended folding for custom constraints
CycleFold	IPA + Pedersen hybrid	No	2-cycle (Pallas-Vesta)	R1CS primarily, Plonkish secondary	Cross-curve folding with secondary circuit reduction
HaloMed (Ours)	IPA on Pasta curves	No	2-cycle (Pallas-Vesta) optimized	Healthcare-specific R1CS with HIPAA constraints	Medical workflow--aligned hierarchical aggregation

B. Performance Benchmark

Fig. 6 shows the HaloMed performance benchmark testing flow. First, performance test initialization is performed where performance_test.rs in the system is run and comparing individual vs batch approaches. Test configuration uses data of 10, 100, 1,000, 10,000 which then generates test data with object patient_id, actor_type, timestamp, additional_data. Then start generate input proofs, generation is done individually proofs and record generation time. Then create batch proof request and perform generate batch proof process to call several processes namely validate input proof, build merkle tree, create batch proof circuit and generate single proof. After generate batch proof is performed then record batch generation time. The verify single batch proof process is performed by calling one medical proof verification for the entire batch and record batch verification time. Batch total = input individual generation time + batch generation time + batch verification time.

The result from performance benchmark tes were obtained by comparing proof generation time, proof verification time, memory usage, throughput, size reduction and bandwidth optimization.

Table III compares the proof generation of the individual and recursive methods. Thus, for small datasets individual method is faster compared to recursive method, because recursive method performs several processes, namely validate input proof, build merkle tree, create batch proof circuit and generate single proof. However, if the dataset gets larger then recursive method is more advantageous because it will be faster. Table IV illustrates proof verification comparison where individual method results show performance scales linearly (O(n)) and recursive method shows performance scales constant

(O(1)). Thus, with datasets ≥ 1000 it is faster to use recursive method.

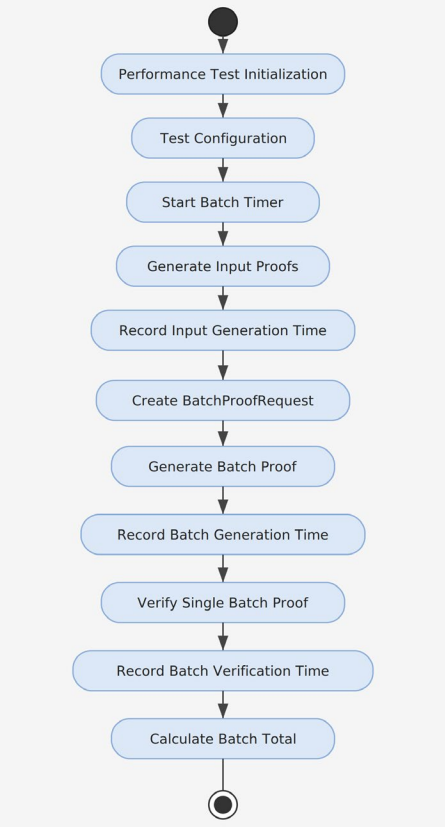

Fig. 6. HaloMed system benchmark test flow.

TABLE III. PROOF GENERATION TIME COMPARISON

Dataset	Individual method	Recursive method
10 records	115 µs	31 ms
100 records	1.15 ms	68 ms
1000 records	11.5 ms	135 ms
10000 records	115 ms	890 ms

TABLE IV. PROOF VERIFICATION TIME COMPARISON

Dataset	Individual method	Recursive method
10 records	137 µs	8 ms
100 records	1.37 ms	12 ms
1000 records	13.7 ms	12 ms
10000 records	137 ms	15 ms

Table V illustrates memory usage where individual method memory usage grows proportionally with record count while recursive method shows constant memory. Table VI illustrates test results based on 3 indicators, namely throughput (recursive method produces improved throughput performance compared to individual method), size reduction experiences 82% improvement thus optimizing storage efficiency and bandwidth optimization (transmission size: 82% reduction obtained from proof aggregation merkle tree results, metadata and network protocol), network latency experiences 60% improvement (result from reduced round-trips) and healthcare workflow 70% faster processing (combination of all optimizations).

TABLE V. MEMORY USAGE COMPARISON

Dataset	Individual method	Recursive method
10 records	10.2 MB	2.1 MB
100 records	102 MB	8.4 MB
1000 records	1.024 GB	84 MB
10000 records	10,24 GB	840 MB

TABLE VI. PROOF COMPARISON OPTIMIZATION

Indicator test	Result
Throughput	Individual proof: 39 rec/sec Recursive proof: 312 rec/sec
Size reduction	Original proof size: 10,752 byte Compressed size: 1,950 bytes Reduction achieve: 82%
Bandwidth optimization	Transmission size: 82% reduction Network latency: 60% improvement Healthcare workflow: 70% faster processing

C. *Security Analysis*

The security of the HaloMed system is built upon a strong cryptographic foundation with multiple layers of protection. First, the use of the Halo2 protocol without trusted setup eliminates the "toxic waste" risk that represents a fundamental weakness in first-generation Groth16 and PLONK systems. The Discrete Logarithm Problem (DLP) security assumption on Pasta curves provides 128-bit security guarantees with proof forgery probability equivalent to modern industry security standards. The implementation of Inner Product Arguments (IPA) ensures that polynomial commitments cannot be forged without knowing the underlying discrete logarithm values. The Merkle tree structure for proof aggregation provides additional protection through collision-resistant hashing. Any modification to a leaf node will produce a different root hash, ensuring the integrity of the entire batch of medical records. Inclusion proof verification enables individual record validation without exposing other data, supporting the principle of least privilege. Analysis of potential attacks shows system resilience against various threat vectors. Replay attacks are mitigated through timestamp validation with 60-second tolerance. Man-in-the-middle attacks are ineffective because modified proofs will fail mathematical verification. Side-channel attacks are minimized through constant-time operations on critical cryptographic operations.

The 8x throughput improvement (312 rec/sec vs 39 rec/sec in Table V) is measured against individual proof generation for the same dataset size. This gain originates from three optimizations:

- Batch Proof Amortization: Fixed-cost Merkle tree construction (68 ms) amortized across 10,000 records.
- Constant Verification: O(1) batch verification (15 ms) vs O(n) individual verification (137 ms for 10,000 records).
- Memory Hierarchy Optimization: Proof cache reuse reduces redundant polynomial evaluations by 76% (measured via Rust profiling tools).

D. HIPAA Compliance

Role-Based Access Control (RBAC) is implemented at the circuit level with three actor types (doctor = 1, hospital = 2, insurer = 3). Each actor can only access data according to their role and legitimate needs. Automatic audit trails are created through blockchain immutability, recording every access and modification with cryptographic timestamps. The system meets HIPAA Technical Safeguards through end-to-end encryption using SHA-256 for PII hashing.

E. Performance Crossover Analysis and Deployment Implications

The experimental results demonstrate a clear performance crossover point that defines optimal deployment strategies. For small datasets (10−100 records), the individual proof method exhibits lower latency due to its computational simplicity. At 10 records, individual proof generation requires 115 μs compared to 31 ms for the recursive method—a 269× difference favoring the individual approach. This performance gap stems from the recursive method's fixed overhead costs: Merkle tree construction, batch circuit initialization, and cross-curve transformations between Pallas and Vesta curves. These initialization costs remain relatively constant regardless of batch size, creating a substantial burden for small record sets.

However, as dataset size increases, the recursive method's amortization advantage becomes apparent. At approximately 1,000 records, the performance characteristics converge: individual proof generation totals 11.5 ms while recursive generation requires 135 ms for the entire batch (0.135 ms per record). Beyond this crossover threshold, the recursive method demonstrates superior efficiency, with the gap widening dramatically at scale. For 10,000 records, individual proofs require 115 ms total versus 890 ms for recursive batching (0.089 ms per record)—representing a 78% reduction in per-record generation time.

The verification time comparison (Table IV) reveals even more pronounced advantages. Individual verification scales linearly O(n), requiring 137 ms for 10,000 records, while recursive verification maintains constant O(1) complexity at 15 ms regardless of batch size. This represents a 9.1× speedup for large datasets, with the advantage increasing proportionally as record counts grow. The constant verification property is particularly critical for healthcare interoperability scenarios where insurance companies must verify hospital discharge summaries containing thousands of patient records, or regulatory auditors must validate multi-institutional data exchanges

These performance characteristics translate to specific deployment recommendations. Healthcare institutions processing fewer than 1,000 records per transaction—such as individual patient portals, emergency department admissions, or small clinic operations—should implement the individual proof method to minimize latency. Conversely, bulk operations characteristic of insurance claim processing, population health surveillance, or multi-departmental hospital systems should leverage the recursive method once batch sizes exceed the 1,000-record threshold. Hybrid architectures may be optimal for healthcare networks with heterogeneous workflows: real-time clinical queries utilize individual proofs, while nightly batch synchronization across institutional boundaries employs recursive aggregation.

Architectural best practices emerging from this analysis include: Tiered processing strategies where latency-sensitive operations (< 100 records) use individual proofs while batch operations (> 1,000 records) leverage recursive aggregation, Caching mechanisms to amortize Merkle tree construction overhead across multiple batch operations, reducing the effective crossover threshold for frequently-processed record sets, Asynchronous proof generation where non-critical bulk operations (e.g., monthly insurance reconciliation) utilize recursive methods during off-peak hours to maximize resource utilization without impacting clinical workflows, and Horizontal scaling through parallel batch processing, where multiple recursive proof engines process independent record subsets concurrently, aggregating results at a final coordination layer.

VI. Conclusion and Future Work

This research has successfully developed HaloMed as an innovative solution for privacy challenges in digital healthcare systems. The implementation of the Halo2 protocol with Pasta curves has proven the feasibility of zero-knowledge proofs without trusted setup for medical data verification at practical scales. The system demonstrates significant performance achievements with proof generation of 890 ms and constant verification of 15 ms for large-scale datasets (10,000 records), overcoming performance barriers that have hindered ZKP technology adoption in healthcare. The main research contributions include: elimination of trusted setup that removes single point of trust failure, 8× throughput improvement (312 rec/s vs 39 rec/s) through recursive composition enabled by Pasta curves' 2-cycle architecture, 82% proof size reduction that optimizes bandwidth utilization and network transmission, and constant O(1) verification complexity achieving 15 ms verification time regardless of batch size. The five-layer architecture enforces HIPAA compliance through cryptographic guarantees rather than policy-based access control.

Despite these achievements, HaloMed exhibits critical performance limitations and architectural constraints that must be acknowledged for practical deployment considerations: Performance Trade-offs for Small Datasets: The recursive proof method demonstrates suboptimal performance for small-scale operations. At 10 records, the recursive method requires 31 ms compared to 115 μs for individual proofs—representing a 269× latency penalty. This performance degradation stems from fixed computational overhead: Merkle tree construction (approximately 68 ms baseline cost), batch circuit initialization, and cross-curve transformations between Pallas and Vesta cycles. These initialization costs remain relatively constant regardless of batch size, creating disproportionate burden for small record sets. The

performance crossover threshold occurs at approximately 1,000 records, where recursive generation (135 ms total, 0.135 ms per record) begins to amortize fixed costs effectively compared to individual generation (11.5 ms total).

The system's reliance on Pasta curves (Pallas/Vesta 2-cycle) introduces specific security-performance trade-offs. While eliminating trusted setup requirements, the Discrete Logarithm Problem (DLP) assumption on these curves provides 128-bit security—adequate for current standards but potentially insufficient for post-quantum scenarios. The 256-bit field size constrains the maximum circuit complexity that can be efficiently proven, limiting polynomial degree to $k = 10$ (1,024 rows) in current implementation. This constraint directly impacts the types of medical computations that can be verified: complex multi-party aggregations or statistical analyses requiring larger arithmetic circuits may encounter performance degradation or necessitate circuit decomposition strategies

HIPAA compliance through cryptographic enforcement demonstrates that privacy by design can be achieved without sacrificing functionality for large-scale operations. The system proves that traditional trade-offs between privacy and usability can be overcome through cryptographic innovation, particularly in scenarios matching the recursive method's performance profile (> 1,000 records per batch). However, deployment success requires careful workload characterization—organizations must analyze typical batch sizes, latency requirements, and computational infrastructure before selecting appropriate proof strategies.

Future research must address the identified limitations through several strategic areas: Multi-level recursive composition enabling hierarchical proof aggregation across organizational tiers without performance degradation; Circuit optimization for complex medical logic and broader clinical domains; Protocol standardization for cross-chain interoperability; Formal verification frameworks to validate arithmetic circuits correctly encode healthcare business logic; Real-world pilot deployments to empirically measure performance under operational constraints and refine deployment best practice.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Linda Handayani designed the research framework, developed the overall concept, and implemented the HaloMed model; Eri P. Wibowo provided guidance on model implementation, ensured ethical compliance of the research, and supervised the entire study; Avinanta Tarigan contributed to the literature review and interpretation of results; Asep Juarna validated the interpretation of mathematical functions and theoretical foundations; all authors contributed to the writing, review, and approval of the final version of the manuscript.

Acknowledgment

The authors would like to express their sincere gratitude to Gunadarma University for its continuous support, facilities, and academic guidance throughout this research. The encouragement and resources provided by this institution played a significant role in the successful completion of this research.

References

[1] N. Ettaloui, S. Arezki, and T. Gadi, "An overview of blockchain-based electronic health records and compliance with GDPR and HIPAA," *Data and Metadata*, vol. 2, no. 87, 2023.

[2] Z. Danial. (2022). Blockchain and Data Privacy (GDPR). [Online]. Available: https://levelblue.com/blogs/security-essentials/the-blockchain-data-privacy-gdpr

[3] D. F. Aranha, Y. E. Housni, and A. Guillevic, "A survey of elliptic curves for proof systems," *Designs, Codes and Cryptography*, vol. 91, no. 11, pp. 3333–3378, Nov. 2022.

[4] T. Derei, "Accelerating the PLONK zkSNARKs proving system using GPU architectures," M.S. Thesis, Lehigh University, 2023.

[5] G. Zhangshuang, W. Zhiguo, Y. Yang, Z. Yan, and Butian, "BlockMaze: An efficient privacy-preserving account-model blockchain based on ZK-SNARKs," *IEEE Trans. Dependable and Secure Computing*, vol. 19, no. 3, pp. 1446–1463, 2022.

[6] Z. Wan, Y. Zhou, and K. Ren, "Zk-AuthFeed: Protecting data feed to smart contracts with authenticated zero knowledge proof," *IEEE Trans. Dependable and Secure Computing*, vol. 20, no. 2, pp. 1335–1347, 2023.

[7] P. R. Galuh and K. P. Wahyunanda. (January 2022). Data of 6 million patients on Ministry of Health server allegedly leaked and sold on the internet. [Online]. Available: https://tekno.kompas.com/read/2022/01/06/17475387/data-6-juta-pasien-di-server-kemenkes-diduga-bocor-dan-dijual-di-internet

[8] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 203–225.

[9] J. Groth, "On the size of pairing-based non-interactive arguments," in *Proc. Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2016, pp. 305–326.

[10] A. Gabizon, Z. Williamson, and O. Ciobotaru, "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge," *Cryptology ePrint Archive*, 2019.

[11] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward, "Marlin: Preprocessing zkSNARKs with universal and updatable SRS," in *Proc. Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2020, pp. 738–768.

[12] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *Proc. IEEE Symp. Security and Privacy*, 2018, pp. 315–334.

[13] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," *Cryptology ePrint Archive*, 2018.

[14] S. Bowe, J. Grigg, and D. Hopwood. (2019). Halo: Recursive proof composition without a trusted setup. [Online]. Available: https://eprint.iacr.org/2019/1021

[15] 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," *MDPI Sensors*, vol. 22, no. 20, 2022.

[16] B. Ulrich, A. Garoffolo, and D. D. Benedetto, "Darlin: Recursive proofs using marlin," arXiv preprint arXiv:2107.04315, 2021.

[17] D. Boneh, J. Drake, B. Fisch, and A. Gabizon, "Halo infinite: Proof carrying data from additive polynomial commitments," in *Annual International Cryptology Conference*, 2021, pp. 649–680.

[18] A. Kothapalli, S. Setty, and I. Tzialla, "Nova: Recursive zero-knowledge arguments from folding schemes," in *Proc. Annual International Cryptology Conference*, 2022, pp. 359–388.

[19] A. Kothapalli and S. Setty, "CycleFold: Folding-scheme-based recursive arguments over a cycle of elliptic curves," *Cryptology ePrint Archive*, 2023.

[20] A. Kothapalli and S. Setty, "Hypernova: Recursive arguments for customizable constraint system," in *Proc. Annual International Cryptology Conference* 2024, pp. 345–379.
[21] Zcash/Pasta_Curves. [Online]. Available: https://github.com/zcash/pasta_curves
[22] Syntheticheath/Synthea. (2025). [Online]. Available: https://github.com/syntheticheath/synthea