Article | Received 10 March 2026; Revised 07 May 2026; Accepted 20 May 2026; Published XX June 2026
https://doi.org/10.55092/blockchain20260004

SST-MedChain: secure and scalable tokenized EMR sharing on a permissioned blockchain

Henglong Zhu, Xiaofei Xing* and Guojun Wang

School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510006, China

* Correspondence author; E-mail: xingxf@gzhu.edu.cn.

Highlights:

- SST-MedChain is formulated as a tokenized authorization state machine for EMR sharing.
- ECDH-based non-interactive delegation and one-time tokens enable lightweight on-chain lookup-and-consume authorization.
- Policy-bounded, traceable re-delegation supports rights conservation and fast revocation via Nested Freezing and a Source Circuit Breaker.

Abstract: Secure and scalable electronic medical record (EMR) sharing is essential for cross-institutional collaboration, yet existing blockchain-based approaches can incur high on-chain overhead under bursty, fine-grained, and temporary authorization. We propose Secure and Scalable Tokenized EMR Sharing on a Permissioned Blockchain, referred to as SST-MedChain, a patient-centric framework that (i) enables patient-side non-interactive delegation via an Elliptic Curve Diffie–Hellman (ECDH)-derived verification token protected by a hash commitment, and (ii) reduces on-chain authorization to a near constant-time token lookup and atomic state transition using one-time access tokens. SST-MedChain further supports policy-bounded cascading re-delegation and fast revocation over deployment-bounded delegation chains via a Nested Freezing state machine and a Source Circuit Breaker. Experiments on FISCO BCOS (a permissioned blockchain platform) in a wide area network (WAN) show that, on the evaluated on-chain confirmation path, SST-MedChain improves throughput by 38% and reduces latency by 86% compared with Attribute-Based Access Control (ABAC) at 300 queries per second (QPS), and achieves 16.5% higher throughput than MedShare at 1000 QPS with comparatively stable average confirmation latency.

Keywords: EMR; blockchain; patient-side non-interactive delegation; one-time tokens; access control

1. Introduction

With the rapid digitization of healthcare, Electronic Medical Records (EMRs) have become central to modern clinical services [1]. In this paper, we use EMR as an umbrella term to refer to electronic

record artifacts commonly called EMR in the literature, without distinguishing them unless explicitly required by a cited scheme. Secure and efficient EMR sharing is increasingly critical for interdisciplinary collaboration and timely decision-making [2]. At the same time, cross-institution sharing raises practical requirements on access governance, auditability, and accountability across organizational boundaries [3].

Permissioned consortium blockchains provide an auditable substrate for trusted cross-institution data circulation, offering tamper-evident logging and traceability under standard Byzantine Fault Tolerance (BFT) assumptions [4]. In particular, their consortium governance and controlled participation make them a natural fit for regulated medical collaboration environments [5]. These properties help mitigate interoperability and trust issues among medical institutions [6].

However, medical access workloads are often bursty and highly concurrent, and permissions are context-dependent. Traditional access control models, such as Attribute-Based Access Control (ABAC), provide expressive and flexible policy specification [7]. In practice, ABAC is often treated as a baseline model for fine-grained authorization in complex domains [8]. When executed on-chain, ABAC-style policy evaluation can require computation-intensive matching over attributes and conditions, which becomes a bottleneck at scale. This overhead can translate into unacceptable latency in time-critical scenarios [9]. Similar observations have been reported when modeling and analyzing healthcare professionals' security practices under attribute-based policies [10].

Furthermore, collaborative diagnosis requires flexible and accountable authorization transfer across staff and institutions. Existing delegation or token-based mechanisms can reduce repeated policy evaluation but may still rely on synchronous interaction [11]. Some token-based designs improve efficiency but provide limited support for auditable, policy-bounded re-delegation and revocation under realistic workflow constraints [12]. These gaps motivate secure and scalable delegation with traceable, constrained re-delegation in healthcare networks [13].

To address these limitations, we propose Secure and Scalable Tokenized EMR Sharing on a Permissioned Blockchain (SST-MedChain), a patient-centric framework that can be abstracted as a tokenized authorization state machine. It combines non-interactive delegation, one-time token authorization, and policy-bounded cascading re-delegation with fast revocation. Specifically, SST-MedChain (i) binds an offline, non-interactive delegation intent to an on-chain record via an Elliptic Curve Diffie–Hellman (ECDH)-derived verification-token commitment, (ii) turns each authorization into a single-use state transition (Unused→Used) under the current owner's signature, and (iii) supports policy-bounded, traceable re-delegation with rights conservation (Nested Freezing) and bounded-cost patient-side revocation (Source Circuit Breaker). Compared with OAuth-style delegation or generic capability systems, SST-MedChain tokens are not replayable bearer credentials: each token is consumable at most once, remains owner-bound on-chain, and can be re-delegated only through a constrained monotone-policy transformation with auditable lineage and bounded rollback.

Our main contributions are summarized as follows:

(1) Non-interactive delegation and one-time authorization: We design an offline delegation mechanism based on ECDH and hash commitments to bind delegation intent without requiring synchronous patient–doctor interaction during request creation. By replacing computation-intensive policy matching with one-time tokens, SST-MedChain reduces on-chain authorization to a near

constant-time logical lookup-and-consume operation, with replay resistance by construction.

(2) Secure cascading re-delegation and bounded-cost revocation: We design a restricted re-delegation protocol based on Nested Freezing and encrypted secret forwarding. This enforces rights conservation to prevent double-spending and introduces a patient-side Source Circuit Breaker for single-transaction revocation along a delegation chain.

(3) Prototype implementation and performance evaluation: Implemented on FISCO BCOS across a wide area network (WAN), SST-MedChain improves throughput by 38% and reduces on-chain confirmation latency by 86% over the ABAC baseline. Under high concurrency, it achieves 16.5% higher throughput than MedShare while maintaining comparatively stable confirmation latency.

Organization: The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the framework overview and system architecture. Section 4 details the design of SST-MedChain. Section 5 analyzes security and complexity. Section 6 reports experimental evaluation. Finally, Section 7 concludes the work.

2. Related work

We review related studies by technical principle: privacy-preserving storage and sharing, fine-grained access control, and delegation/token-based authorization.

Privacy-preserving storage and sharing: To address data sovereignty and privacy risks in cross organization sharing, researchers have explored architectures that combine decentralized storage with edge-assisted enforcement. Shahzad *et al.* [14] proposed a zero-trust medical image sharing approach, utilizing edge nodes to forward requests and enabling patients to manage permissions directly using Elliptic Curve Cryptography (ECC) and Advanced Encryption Standard (AES) encryption. Focusing on storage efficiency, Li *et al.* [15] designed "EHRChain," which integrates InterPlanetary File System (IPFS) for high-capacity storage and employs an improved SHDPCPC-CP-ABE algorithm (a ciphertext-policy attribute-based encryption (CP-ABE) variant) to achieve semi-privacy policy hiding. Sun *et al.* [16] constructed a secure storage scheme based on Hyperledger and IPFS, aiming to improve scalability while keeping sensitive EMR data off-chain.

Fine-grained access control: Fine-grained authorization on blockchain often relies on cryptographic primitives to reduce trust assumptions while preserving auditability. For Internet of Medical Things (IoMT) data publishing, Wu *et al.* [17] developed a purpose-aware access model that enforces purpose legitimacy and incorporates local differential privacy. Zhao *et al.* [18] introduced a blockchain-edge collaboration strategy, combining proxy re-encryption (PRE) with verifiable anonymization to ensure accountability alongside fine-grained control. Wang *et al.* [19] proposed "MedShare," a trusted sharing framework that employs an Attribute-Based Encryption (ABE) mechanism to support multi-keyword Boolean search for Electronic Health Record (EHR) sharing and verification. Recent work has further emphasized risk-adaptive authorization in decentralized IoT ecosystems. In the healthcare domain, Chu *et al.* [20] integrated smart contracts with Role-Based Access Control (RBAC) and real-time access behavior analysis to enable secure, scalable, and high-frequency access management for large-scale EMR systems.

Delegation and token-based authorization: Beyond enforcing static policies, clinical collaboration often requires accountable delegation and revocation across parties. Mukta *et al.* [21] proposed a zero-trust-driven delegation architecture that combines Self-Sovereign Identity (SSI)/Decentralized Identifier (DID)-based identity with capability-based access control and continuous trust evaluation, while leveraging blockchain to provide tamper-evident auditing and accountability. Alshehri *et al.* [22] utilized Hyperelliptic Curve Cryptography (HECC) to generate key pairs, employing a Rock Hyrax Swarm Optimization (RHSO) based access delegation selection algorithm and a Practical Byzantine Fault Tolerance (PBFT)-based permissioned blockchain to provide consistent logging and accountable enforcement. Said *et al.* [23] adopted a method combining the Hyperledger Fabric blockchain platform with Non-Fungible Token (NFT) technology to achieve patient control over health records and temporary authorized access to medical professionals. Furthermore, Saha *et al.* [24] realized secure communication between patients and medical management agencies, as well as the secure exchange of medical information among providers, through the use of smart contracts in 6G-enabled systems. Duan *et al.* [25] proposed a multi-authority Attribute-Based Encryption (ABE) scheme with delegated access for cross-blockchain data sharing, supporting ciphertext-policy updates through a proxy-based workflow. Their design leverages relay-chain coordination and smart contracts to automate cross-chain sharing while reducing on-chain storage overhead.

Despite this progress, prior work often addresses only part of the problem: off-chain storage reduces ledger bloat but leaves high-frequency authorization costly; expressive models incur heavy on-chain verification; and token/delegation mechanisms may require interaction and offer limited support for auditable, policy-bounded cascading re-delegation with practical revocation under depth constraints. SST-MedChain targets the clinical "high concurrency + temporary authorization transfer" setting by combining offline delegation, efficient token consumption, and restricted cascading re-delegation with revocation via Nested Freezing and a Source Circuit Breaker.

3. Framework overview

3.1. System model

Figure 1 illustrates the SST-MedChain architecture, which separates the control plane (on-chain authorization) from the data plane (off-chain storage). In SST-MedChain, the blockchain primarily serves as an auditable control-plane trust layer that supports atomic authorization state transitions and tamper-evident logging; it does not execute heavy cryptography over EMR contents, which remain off-chain. Conceptually, similar control-plane functionality could be implemented using a consortium-operated replicated database with append-only logs; we adopt a permissioned blockchain to align with consortium governance, deterministic execution, and tamper-evident auditability under standard BFT assumptions, instantiated as PBFT in our prototype deployment. The system comprises four key entities:

- Patient & Medical Staff: Patients act as data owners and create offline delegation requests (with a verification-token commitment). These signed requests are sent to hospital nodes, which submit them on-chain. Medical staff (Doctors) complete on-chain verification and one-time tokens registration via hospital nodes, and then consume tokens to access EMRs or initiate restricted

re-delegation for consultations.

- Hospital Nodes: Acting as semi-honest agents, these permissioned nodes facilitate user interactions and relay transactions. Users (patients/doctors) authorize every operation by signing the request payload under their registry-bound keys, and hospital nodes submit the signed payloads on-chain. Smart contracts verify signatures against the on-chain registries (Subject Management Contract (SMC)/Object Management Contract (OMC)) before updating authorization state. Hospital nodes handle computationally intensive tasks (e.g., encryption/decryption) and maintain a local cache to reduce query latency.
- Blockchain Network: A permissioned consortium blockchain serves as the control plane. It maintains the global permission state (token state machines) and records access logs for accountability.
- InterPlanetary File System: Provides content-addressed storage for EMR ciphertext (e.g., a consortium-operated private IPFS swarm) encrypted under a symmetric key K_{AES} using Advanced Encryption Standard in Galois/Counter Mode (AES-GCM) (e.g., AES-128-GCM). For each EMR encryption under a fixed K_{AES}, the initialization vector IV_{emr} is sampled uniformly at random (96-bit) and never reused. Each EMR (or access scope) is associated with a distinct K_{AES} generated and securely stored at the data-holding hospital (e.g., in a Key Management Service/Hardware Security Module (KMS/HSM)-backed keystore). Only the Content Identifiers (CIDs) are anchored on-chain to ensure lightweight ledger operations; K_{AES} is never posted on-chain and is released to requesters only after on-chain authorization and target-side verification.

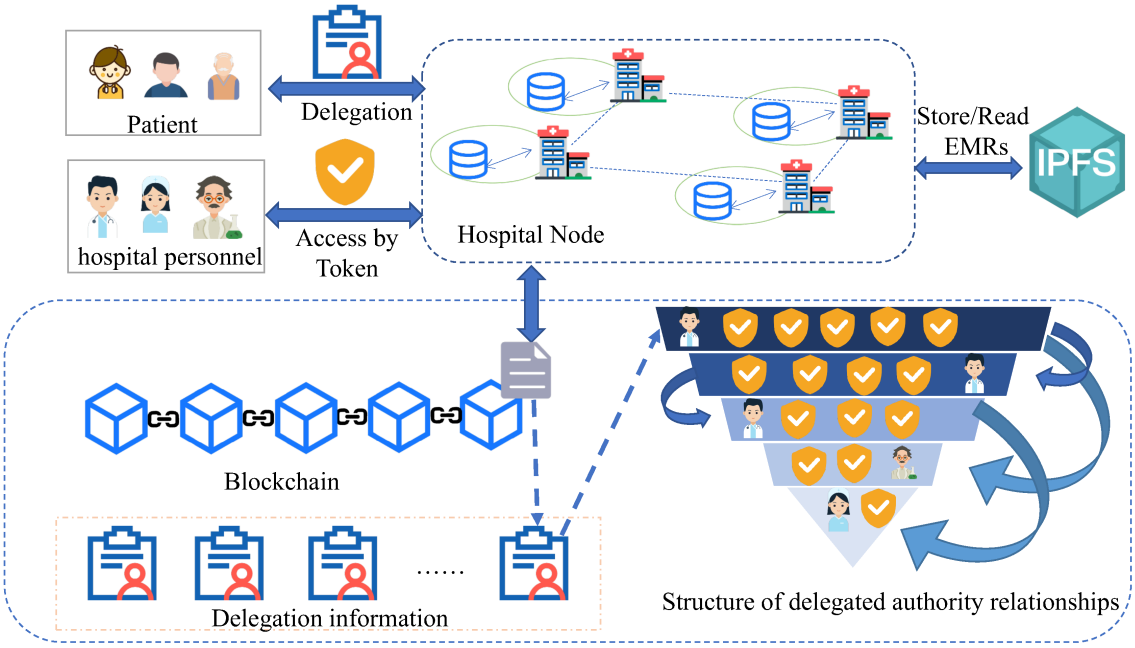

Figure 1. System framework.

3.2. Smart contract architecture

We adopt a "Logic-Data Separation" pattern: the orchestration contract (Request Decision Contract (RDC)) executes decision logic, while the sub-contracts (Delegation Management Contract (DMC)/SMC/OMC) store on-chain state and expose interfaces invoked by the RDC. As shown in Figure 2, the architecture

consists of one orchestration contract (RDC) and three sub-contracts (DMC, SMC, and OMC):

- Request Decision Contract: The system's central gateway and logic engine. It exposes unified interfaces to external nodes and orchestrates the sub-contracts to execute atomic access decisions.
- Delegation Management Contract: Manages the lifecycle of permissions. It maintains a mapping from token identifiers ($tid = H(w)$) to token states (Unused, Used, Frozen, Revoked) and enforces policy constraints (e.g., expiration, allowed operations), serving as the state machine for the token system.
- Subject Management Contract: Acts as the identity registry for consortium participants. It binds consortium-scoped identifiers for medical staff (PID_D) to public keys and (optional) attribute metadata (e.g., department, role). It also binds hospital node identifiers (HID_H) to hospital public keys PK_H for cross-hospital key encapsulation and request attribution.
- Object Management Contract (OMC): Manages patient-side records and EMR indices. It maintains patient basic information and public keys, and maps $(PID_P, AccessScope)$ to IPFS CIDs, allowing the RDC to resolve EMR locations upon successful authorization.

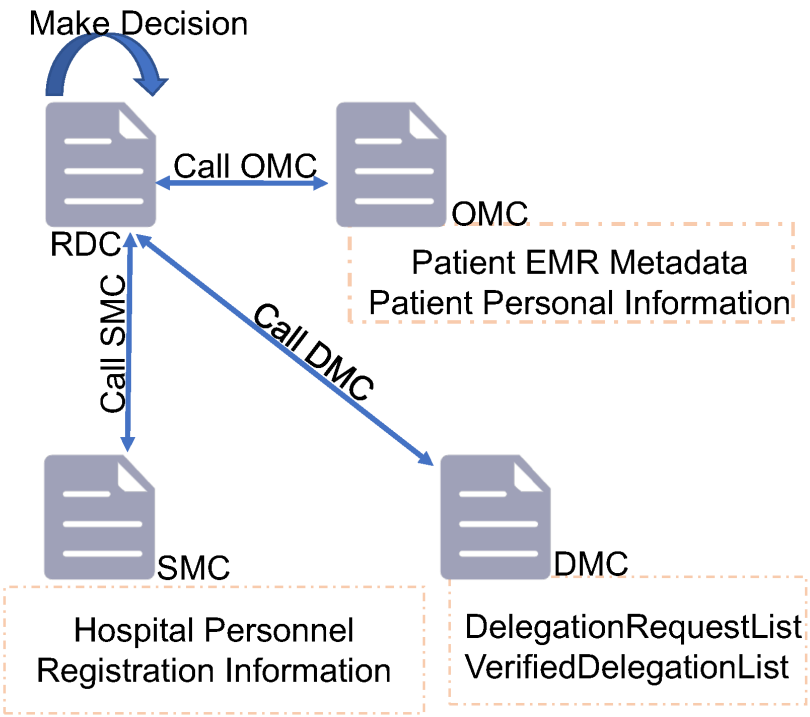

Figure 2. Smart contract relationship structure diagram.

4. Design of SST-MedChain

This section presents SST-MedChain. Section 4.1 describes a Non-Interactive Delegation Protocol for offline delegation authentication and one-time token registration with near constant-time logical lookup-and-consume checks. Section 4.2 presents policy-bounded cascading re-delegation and revocation. Figure 3 illustrates the delegation workflow. Table 1 summarizes notations.

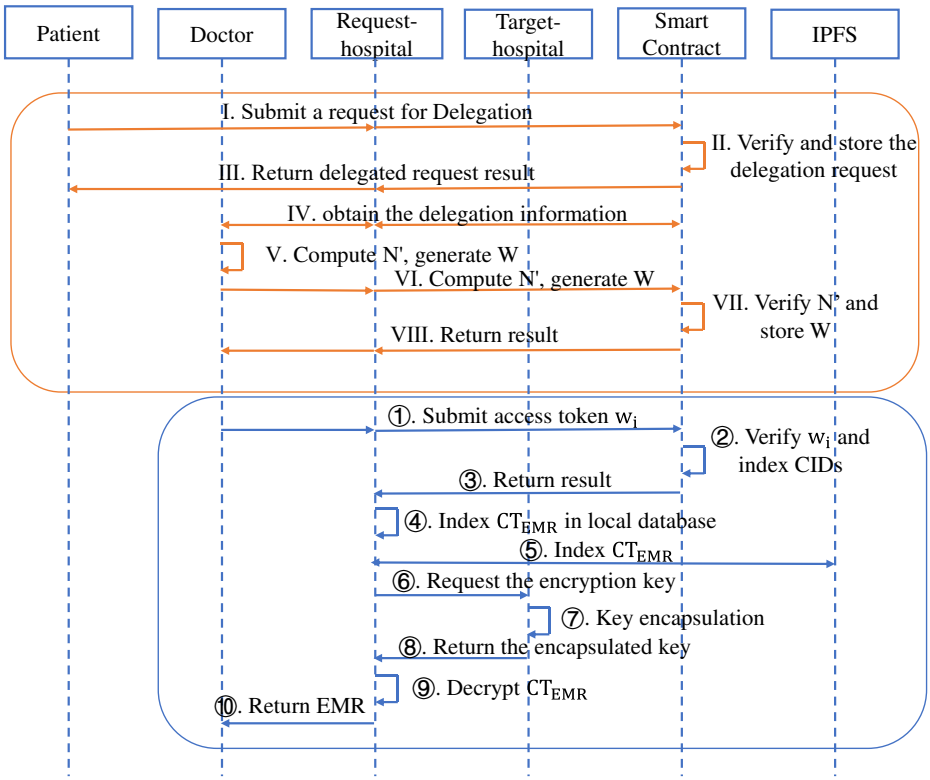

Figure 3. Delegated access control process.

Table 1. Symbol definitions.

Symbol	Definition
SK_i, PK_i	The private key and public key of entity i
M, σ	Delegation Request Message and its Digital Signature
$\mathscr{P}$	Delegation parameters
r	Randomisation factor
S	Shared Secret
N, c	Verification token and its hash commitment
$H(\cdot)$	Cryptographic hash function (SHA-256)
w_i, s_i	One-time token value (encoded curve point) and its corresponding secret
$\text{Encode}(\cdot), \text{Decode}(\cdot)$	Canonical point encoding/decoding on SECP256k1; Encode outputs Standards for Efficient Cryptography (SEC1) compressed bytes and Decode rejects invalid/non-canonical points
tid_i	Token identifier stored on-chain, $tid_i = H(w_i)$ (e.g., bytes32 via sha256)
did	Delegation identifier computed as $did = H(M)$ and used for patient-side revocation
$\mathscr{W}$	Set of One-time Tokens
HID_H	Consortium-scoped identifier of a hospital node (requester)
PK_H	Public key of the requesting hospital node, bound to HID_H
PK_{Com}	Combined public key for secure key encapsulation, $PK_{Com} = PK_H + \text{Decode}(w_i)$
K_{AES}	Symmetric key for encrypting medical record data
$CT_{K_{AES}}$	Key encapsulation packet
IV	Fresh 96-bit initialization vector for AES-GCM; it must be unique for each encryption under a fixed key
HKDF-SHA256	HKDF instantiated with HMAC-SHA256 (Extract-and-Expand KDF)
CID*	Target descriptor returned after authorization; in the prototype it is a single CID, while the same interface naturally extends to a canonical CID list
$\mathbb{G}, g$	Cyclic group of order q and its generator
C	Encapsulation ciphertext component
d_{max}	Configurable maximum delegation depth threshold

4.1. Non-interactive delegation protocol

Delegation Request Generation: Let $\mathbb{G}$ be a cyclic group of prime order q with generator g, and $PK_i = SK_i \cdot g$. Patient P defines a policy $\mathscr{P} = \{AccessScope, Ops, T_{exp}, Times\}$ with token count $Times = n$. In particular, T_{exp} denotes the authorization validity deadline of the delegation policy. Here, Ops denotes the set of permitted operations associated with the delegation (e.g., $Ops = \{read\}$ for read-only access, or $Ops = \{read, redelegate\}$ to explicitly allow permission sharing), depending on deployment semantics. Therefore, re-delegation is not an implicit capability of every delegatee; it is permitted only when the parent delegation policy explicitly includes the corresponding re-delegation permission in Ops. P samples $r \in \mathbb{Z}_q$, computes $R = r \cdot g$, and derives a shared secret using PK_D:

$$S = r \cdot PK_D = r \cdot (SK_D \cdot g) \tag{1}$$

P computes $N = \text{HKDF-SHA256}(\text{Encode}(S))$ and commitment $c = H(N)$, constructs $M = (PID_P, PID_D, R, c, \mathscr{P}, Timestamp)$, and signs $H(M)$ to obtain σ. A hospital node submits (M, σ) to the RDC, which validates PID_D (SMC) and PID_P (OMC), derives $did = H(M)$, and stores $DelegationRequestList[did]$ in the DMC. To mitigate replay of stale delegation intents and tolerate practical clock skew, the contract validates Timestamp against the chain time with an acceptance window Δ (e.g., 5 minutes), rejecting requests outside $[t_{chain} - \Delta,\ t_{chain} + \Delta]$.

On-Chain Verification and Token Initialization: Doctor D reconstructs the shared secret using SK_D:

$$S' = SK_D \cdot R = SK_D \cdot (r \cdot g) = r \cdot PK_D = S \tag{2}$$

D recovers $N' = \text{HKDF-SHA256}(\text{Encode}(S'))$ and checks $H(N') = c$. Then D generates k ($k \leq n$) one-time token secrets $\{s_i\}_{i=1}^{k}$ and token points $\mathscr{W} = \{w_i \leftarrow \text{Encode}(s_i \cdot g)\}$. The contract binds $(\mathscr{P}, PID_D)$ to did and registers the submitted one-time tokens for later consumption. D signs $(did, N', \mathscr{W})$ and a hospital node submits the request; the RDC executes VerifyDelegation():

(1) Verifies the signature to ensure the request is authorized by D.

(2) Verifies $H(N') = c$ to prove correct secret derivation.

(3) Finalizes the delegation record on-chain and registers the submitted one-time tokens as Unused.

Access rights are bound to one-time tokens.

Token-based Data Access and Target Verification: The doctor requests access with (s_i, ctx, op) authorized by the bound PID_D, where $ctx = (PID_P, AccessScope_{req})$. The hospital node derives $w_i = \text{Encode}(s_i \cdot g)$ and forwards a signed access request $(w_i, ctx, op, HID_H, \sigma_{acc})$ to the RDC.

The contract derives a token identifier from w_i, checks that it is Unused and that the request satisfies $\mathscr{P}$, verifies σ_{acc} under the current token owner's public key, then atomically marks it as Used and returns the bound target descriptor.

Upon receiving the bound target descriptor, the hospital node retrieves the corresponding EMR ciphertext CT_{EMR} from the consortium IPFS deployment and caches it locally for decryption. For *local* EMRs where the data-holding hospital is the requester itself, the hospital releases the corresponding K_{AES} from its local secure keystore only after the on-chain authorization succeeds, and then decrypts CT_{EMR} locally.

To facilitate subsequent verification and bind the retrieval target to the correct requester hospital key, the contract computes and stores a binding digest $bind_{on} = H(tid, ctx, op, HID_H, H(\text{CID}^*))$. This lookup-and-update process is near constant-time in the size of the policy/attribute space (logical $O(1)$ token lookup and state transition), excluding consensus latency and hashing/encoding work proportional to the size of the returned target descriptor. Here, $H(\text{CID}^*)$ is computed over a canonical representation of the target descriptor.

Authorization is enforced by signature verification: the current token owner signs the access request payload, and the RDC validates the signature against the owner's public key obtained from the SMC registry via $DidOf[tid]$ before consuming the token.

For remote data, the retrieval includes a Target Verification Step:

(1) Verification Request: The requesting hospital node sends $(w_i, ctx, op, HID_H, \text{CID}^*)$ to the target hospital node.

(2) On-Chain Check: The target recomputes the binding digest and submits it on-chain; the contract checks that the token has been consumed and that the stored digest matches.

Only upon verification does the target proceed. The target hospital retrieves the corresponding K_{AES} from its secure keystore and releases it to the requester only after the on-chain verification succeeds. To protect K_{AES} in transit, we use a token-bound Key Encapsulation Mechanism (KEM). The target then resolves the *requesting hospital*'s public key PK_H from HID_H (as committed in $bind_{on}$), and encapsulates K_{AES} to $PK_{Com} = PK_H + \text{Decode}(w_i)$, so decryption requires both SK_H and s_i. After decapsulation, the requesting hospital decrypts the cached CT_{EMR} fetched via the bound target descriptor and discards K_{AES} after use.

The target hospital never learns s_i: it only receives $w_i = \text{Encode}(s_i \cdot g)$, and recovering s_i from w_i is exactly the elliptic-curve discrete logarithm problem in $\mathbb{G}$.

Protocol Requirement (Canonical Encoding and Point Validation): w_i is treated as a byte string on-chain (hashed to derive $tid_i = H(w_i)$) but is decoded as a curve point off-chain for KEM. All parties must use a canonical encoding for w_i and a strict decoder. We instantiate $\text{Encode}(\cdot)$ as SEC1 compressed point encoding on SECP256k1 and require that $H(\cdot)$ is computed over this canonical byte representation. When computing $PK_{Com} = PK_H + \text{Decode}(w_i)$, the target hospital must validate both PK_H and $\text{Decode}(w_i)$ as curve points, and reject any w_i whose decoding fails (wrong length/prefix), is not on the curve, or represents the point at infinity. In addition, it must reject the degenerate case $PK_{Com} = \mathcal{O}$ (the point at infinity) to avoid trivial or ill-defined shared secrets.

Symmetric Key Encapsulation: A fresh random $r \leftarrow \mathbb{Z}_q$ is generated to compute $R = r \cdot g$. The shared secret is derived as:

$$S = r \cdot PK_{Com} = r \cdot (PK_H + \text{Decode}(w_i)) = r \cdot (SK_H + s_i) \cdot g \tag{3}$$

A session key is derived as:

$$K_{session} = \text{HKDF-SHA256}(\text{Encode}(S)) \tag{4}$$

For each encapsulation, the target hospital samples a fresh 96-bit IV uniformly at random and never reuses it under the same $K_{session}$. To encapsulate K_{AES} via AES-GCM:

$$(C, Tag) = \text{AES-GCM\_enc}(K_{session}, IV, K_{AES}) \tag{5}$$

The encapsulation packet is:

$$CT_{K_{AES}} = (R, IV, C, Tag) \tag{6}$$

The package in Equation (6) is sent to the requesting hospital node, which reconstructs $S' = (SK_H + s_i) \cdot R$, derives the session key (Equation (4)), and decrypts K_{AES}. The EMR is then decrypted and K_{AES} is discarded after use. The complete flow is detailed in Algorithm 1.

Algorithm 1 NIDP: Non-Interactive Delegation Protocol

Input: Policy $\mathscr{P}$; patient delegation intent $(PID_P, PID_D, \mathscr{P})$; access request context (ctx, op) where $ctx = (PID_P, AccessScope_{req})$; requesting hospital identifier HID_H; (off-chain) token secret s_i; (off-chain) signing key of current token owner SK_{owner}
Output: Grant/Deny and (if granted) the bound target descriptor and/or decrypted EMR
Patient (off-chain): $r \leftarrow \mathbb{Z}_q$; $R \leftarrow r \cdot g$; $N \leftarrow$ HKDF-SHA256(Encode$(r \cdot PK_D)$); $c \leftarrow H(N)$
$M \leftarrow (PID_P, PID_D, R, c, \mathscr{P}, Timestamp)$; $\sigma_P \leftarrow \text{Sign}(SK_P, H(M))$
Send (M, σ_P) to hospital node; hospital node submits to RDC
Authentication (on-chain): $PK_D \leftarrow SMC.GetPK(PID_D)$; $PK_P \leftarrow OMC.GetPK(PID_P)$; $did \leftarrow H(M)$
if not Verify$(PK_P, H(M), \sigma_P)$ **or** timestamp invalid **then**
    Reject
end if
$DMC.DelegationRequestList[did] \leftarrow \langle c, \mathscr{P}, R, PID_P, PID_D \rangle$
Enrollment (doctor off-chain+on-chain): $N' \leftarrow$ HKDF-SHA256(Encode$(SK_D \cdot R)$)
if $H(N') \neq c$ **then**
    Reject
end if
$\mathscr{W} \leftarrow \{\text{Encode}(s_i \cdot g)\}_{i=1}^{k}$; $\sigma_D \leftarrow \text{Sign}(SK_D, H(did, N', \mathscr{W}))$
Send $(did, N', \mathscr{W}, \sigma_D)$ to hospital node; hospital node submits to RDC; RDC verifies Verify$(PK_D, H(did, N', \mathscr{W}), \sigma_D)$ and $H(N') = c$
$DMC.VerifiedDelegationList[did] \leftarrow \langle PID_P, PID_D, \mathscr{P} \rangle$
for all $w_i \in \mathscr{W}$ **do**
    $tid_i \leftarrow H(w_i)$; $DMC.Register(tid_i, did)$; $Roots[did].append(tid_i)$
end for
Consume (owner authorized, submitted on-chain): $w \leftarrow \text{Encode}(s_i \cdot g)$; $\sigma_{acc} \leftarrow \text{Sign}(SK_{owner}, H(w, ctx, op, HID_H))$; send $(w, ctx, op, HID_H, \sigma_{acc})$ to hospital node; hospital node submits to RDC; $tid \leftarrow H(w)$
$(PID_P, AccessScope_{req}) \leftarrow ctx$
$PID_{owner} \leftarrow VerifiedDelegationList[DidOf[tid]].PID_D$; $PK_{owner} \leftarrow SMC.GetPK(PID_{owner})$
if $Status[tid] \neq Unused$ **or** policy/expiry mismatch **or not** Verify$(PK_{owner}, H(w, ctx, op, HID_H), \sigma_{acc})$ **then**
    Deny
end if
$DMC.Consume(tid)$
$\text{CID}^* \leftarrow OMC.Query(PID_P, AccessScope_{req})$
$CT_{EMR} \leftarrow IPFS.Get(\text{CID}^*)$
$bind_{on} \leftarrow H(tid, ctx, op, HID_H, H(\text{CID}^*))$; $DMC.Bind[tid] \leftarrow bind_{on}$
if EMRs are local **then**
    Retrieve K_{AES} from local secure keystore
    Decrypt CT_{EMR} and return EMR; discard K_{AES}
else
    Requesting hospital node sends $(w, ctx, op, HID_H, \text{CID}^*)$ to target hospital node
    Target hospital node checks that Decode(w) succeeds and is a valid SECP256k1 point
    Target hospital node computes $\hat{bind} \leftarrow H(tid, ctx, op, HID_H, H(\text{CID}^*))$ and submits $(tid, \hat{bind})$ to RDC
    RDC verifies $Status[tid] == \texttt{Used} \wedge DMC.Bind[tid] == \hat{bind}$; if valid, Target resolves PK_H from HID_H and returns $CT_{K_{AES}}$ to $PK_H +$ Decode(w)
    Requester decrypts K_{AES} and then decrypts cached CT_{EMR}; discard K_{AES}
end if

4.2. Cascading restricted re-delegation

In consultations, a physician (D_i) may need to delegate part of the remaining access count to a specialist (D_j). SST-MedChain re-delegates by freezing the delegator's unused tokens on-chain, activating the same tokens for the delegatee under a tightened policy, and forwarding the corresponding token secrets off-chain encrypted under the delegatee's public key. To ensure traceability and enable permission rollback upon revocation, the new delegation record explicitly includes the upper-level delegation identifier (did_n).

Each re-delegation transfers a subset of the delegator's remaining count while requiring a stricter policy. To limit risk from long delegation chains, each level ($D_n \to D_{n+1}$) must satisfy:

- Policy Subset Constraint: $\mathscr{P}_{n+1} \subseteq \mathscr{P}_n \subseteq \cdots \subseteq \mathscr{P}_{original}$. Specifically, the child delegation must remain bounded by the parent in access scope ($AccessScope$), permitted operations (Ops), and validity period (T_{exp}). In particular, a child delegation can be created only if the parent policy explicitly permits re-delegation through Ops; otherwise, the re-delegation request is rejected.
- Token Flow Conservation: The access count (number of tokens) obtained by the lower-level delegatee must directly originate from the upper-level's unused share: each transferred token is first locked ($Unused \to Frozen$) by the delegator and then re-activated ($Frozen \to Unused$) for the delegatee with a tighter policy, keeping the same identifier while preserving traceability across levels.

The DMC maintains a per-token lifecycle (Unused, Used, Frozen, Revoked) and supports recovery of still-unused tokens by rolling back the current child delegation binding to the parent delegation record and, where applicable, releasing tokens from the frozen state.

To balance clinical flexibility and audit tractability, the system supports a configurable maximum delegation depth threshold d_{max}. In practice, shorter delegation chains reduce abuse surface, simplify incident response, and keep lineage traversal manageable.

Taking a typical three-level consultation scenario as an example: The attending physician D_1 delegates to expert D_2, and expert D_2 further delegates to assistant D_3 to organize data.

Stage 1: Second-level Delegation ($D_1 \to D_2$). D_1 selects a subset $\mathscr{W}_{sub1} \subset \mathscr{W}_1$ and tightens the policy to $\mathscr{P}_2$. Off-chain, D_1 encrypts the corresponding token-secret subset $\{s_i\}$ under the delegatee's public key PK_{D_2} (e.g., a standard public-key encryption / hybrid encryption scheme) and sends the ciphertext to D_2, who decrypts it with SK_{D_2} to recover $\{s_i\}$. The ciphertext is bound to the delegation context $(PID_P, D_2, \mathscr{P}_2, T_{exp})$ via authenticated associated data under a canonical byte encoding. On-chain, the DMC atomically freezes the selected tokens ($Unused \to Frozen$) and activates the same tokens for D_2 by binding them to a new delegation record with $(PID_P, D_2, \mathscr{P}_2)$.

Secret-forwarding encryption: We instantiate the off-chain secret forwarding as a standard hybrid public-key encryption with authenticated encryption with associated data (AEAD). Concretely, D_n forms a payload $m = \langle (tid_1, s_1), \ldots, (tid_m, s_m) \rangle$ for the transferred subset and defines associated data ad under a canonical encoding as:

$$ad = H(PID_P, PID_{D_{n+1}}, did_{new}, \mathscr{P}_{n+1}, T_{exp}) \tag{7}$$

Then D_n samples an ephemeral scalar $r_f \leftarrow \mathbb{Z}_q$, computes $R_f = r_f \cdot g$ and a shared secret $S_f = r_f \cdot PK_{D_{n+1}}$, derives $K_f = \text{HKDF-SHA256}(\text{Encode}(S_f))$, samples a fresh 96-bit IV_f uniformly at random, and

outputs $CT_{fwd} = (R_f, IV_f, C_f, Tag_f)$ where $(C_f, Tag_f) = \text{AES-GCM\_enc}(K_f, IV_f, m; ad)$. The delegatee D_{n+1} derives $S'_f = SK_{D_{n+1}} \cdot R_f$, reconstructs K_f, recomputes $ad' = H(PID_P, PID_{D_{n+1}}, did_{new}, \mathscr{P}_{n+1}, T_{exp})$ from the accepted child-delegation context, and then computes

$$K'_f = \text{HKDF-SHA256}(\text{Encode}(S'_f)), \quad \hat{m} = \text{AES-GCM\_dec}(K'_f, IV_f, C_f, Tag_f; ad'). \tag{8}$$

If decryption returns $\perp$, the ciphertext is rejected. Thus, acceptance requires both a valid authentication tag and a match between the sender's associated data and the delegatee's recomputed context binding ad'. Otherwise, the delegatee parses $\hat{m}$ as $\langle (tid_1, s_1), \ldots, (tid_m, s_m) \rangle$ and validates each tuple by checking $tid_i = H(\text{Encode}(s_i \cdot g))$ before accepting the forwarded subset.

The construction satisfies the standard ECDH correctness relation

$$S_f = r_f \cdot PK_{D_{n+1}} = r_f \cdot (SK_{D_{n+1}} \cdot g) = SK_{D_{n+1}} \cdot R_f = S'_f, \tag{9}$$

so both parties derive the same forwarding key

$$K_f = \text{HKDF-SHA256}(\text{Encode}(S_f)) = \text{HKDF-SHA256}(\text{Encode}(S'_f)). \tag{10}$$

Moreover, because the AEAD context is bound to the delegation metadata, a ciphertext is accepted only if the delegatee recomputes ad' from the accepted child-delegation record; any mismatch from the sender's encryption-time associated data causes AEAD verification failure. This prevents re-binding the forwarded token-secret bundle to a different delegatee, delegation identifier, policy, or expiry context.

Stage 2: Third-level Delegation ($D_2 \to D_3$). D_2 repeats the procedure with $\mathscr{W}_{sub2} \subseteq \mathscr{W}_{sub1}$ and $\mathscr{P}_3 \subseteq \mathscr{P}_2$.

Stage 3: Delegatee Data Access (D_3 Execution). D_3 decrypts the received secrets and consumes tokens via the same access interface as the first-level delegation; cross-hospital retrieval follows Section 4.1.

Stage 4: Permission Flow-back and Traceability. Revocation supports step-by-step rollback or cross-level circuit breaking:

(1) Step-by-step Recovery: Unused tokens currently bound to a child delegation can be returned one level up by rolling back activation to the parent delegation record, authorized by the immediate parent delegator.

(2) Source Circuit Breaker: The root party revokes by referencing the root delegation identifier $did = H(M)$ (or, for partial rollback, directly referencing a token identifier tid). The contract maintains an index $Roots[did]$ that records the token identifiers issued under the root delegation. A single revocation transaction iterates over $Roots[did]$ and invalidates all unconsumed tokens (regardless of their current delegation level), without requiring the current token owner's signature.

To keep the Source Circuit Breaker feasible under per-transaction resource limits, the system bounds the token count $Times$ at delegation issuance time (hence $|Roots[did]| \leq Times$) and requires the realized delegation depth to remain within the configured threshold. Revocation prevents subsequent authorization and key release for future requests, but it cannot retract EMR data that has already been disclosed and decrypted before the revocation transaction is confirmed.

Algorithm 2 summarizes the on-chain lifecycle management for multi-level permissions, including Nested Freezing state transitions and the "Source Circuit Breaker" revocation.

Algorithm 2 CRR-SCB: Cascading Restricted Re-delegation and Source Circuit Breaker Revocation

Input: Delegator D_n, delegatee D_{n+1}, delegation identifier of delegator did_n, transfer subset $\mathscr{W}_{tr} \subset \mathscr{W}_n$, tightened policy $\mathscr{P}_{n+1}$, (optional) revoke root identifier tid_{root}, (optional) delegation identifier did, current depth $depth_n$
Output: On-chain state updates for identifiers derived from $\mathscr{W}_{tr}$ (and if invoked) revocation result
$\mathscr{P}_n \leftarrow VerifiedDelegationList[did_n].\mathscr{P}$
if CanRedelegate$(\mathscr{P}_n.Ops) = 0$ **then**
    Re-delegation Not Permitted
end if
if $\mathscr{P}_{n+1} \not\subseteq \mathscr{P}_n$ **then**
    Invalid
end if
Re-delegation (CRR):
if $depth_n + 1 > d_{max}$ **then**
    Depth Exceeded
end if
Atomic freeze-and-activate (authorized by D_n):
for all $w \in \mathscr{W}_{tr}$ **do**
    $tid \leftarrow H(w)$; $did_{prev} \leftarrow DidOf[tid]$
    if $Status[tid] \neq Unused$ **or** $did_{prev} \neq did_n$ **or** $VerifiedDelegationList[did_{prev}].PID_D \neq D_n$ **or** transaction not authorized by D_n **then**
        Invalid
    end if
    set $Status[tid] : Unused \rightarrow Frozen$
    $PID_P \leftarrow VerifiedDelegationList[did_{prev}].PID_P$
    $did_{new} \leftarrow H(tid, D_{n+1}, \mathscr{P}_{n+1}, Timestamp)$; $PrevDid[did_{new}] \leftarrow did_{prev}$
    $VerifiedDelegationList[did_{new}] \leftarrow \langle PID_P, D_{n+1}, \mathscr{P}_{n+1} \rangle$; $DidOf[tid] \leftarrow did_{new}$
    set $Status[tid] \leftarrow Unused$
end for
Rollback/Revocation:
if did is provided **or** tid_{root} is provided **then**
    $PID_{root} \leftarrow$ Patient bound to did/tid_{root}
    $PID_{delegator} \leftarrow$ Doctor bound to $PrevDid[did]$ (if exists)
    if tx not authorized by PID_{root} **then**
        if not (did provided **and** tx authorized by $PID_{delegator}$) **then**
            Unauthorized
        end if
    end if
    if did is provided **then**
        $did_{root} \leftarrow$ follow $PrevDid[\cdot]$ from did to the root (at most d_{max} steps)
        $\mathscr{X} \leftarrow Roots[did_{root}]$
    else
        $\mathscr{X} \leftarrow \{tid_{root}\}$
    end if
    for all $x \in \mathscr{X}$ **do**
        if $Status[x] \in \{Unused, Frozen\}$ **then**
            if tx authorized by PID_{root} **then**
                $Status[x] \leftarrow Revoked$
            else if did is provided **and** tx authorized by $PID_{delegator}$ **and** $PrevDid[did]$ is defined **and** $DidOf[x] == did$ **then**
                $DidOf[x] \leftarrow PrevDid[did]$
            end if
        end if
    end for
end if

5. Security and complexity analysis

This section analyzes the security of SST-MedChain under the stated threat model and evaluates the computational complexity of the core algorithms.

5.1. Threat model

Let λ denote the security parameter. We instantiate $\mathbb{G}$ on SECP256k1 and use SHA-256 for H (on-chain via sha256). We use ECDSA for signatures, HKDF-SHA256 for key derivation, and AES-GCM for authenticated encryption, with the main notation summarized in Table 1.

All uses of HKDF take a canonical byte string derived from group operations. For a shared secret point $S \in \mathbb{G}$, we instantiate HKDF as HKDF-SHA256(Encode(S)) using the same SEC1 compressed encoding as in Section 4.1. All hashes and signatures over structured objects are computed over an unambiguous canonical byte encoding.

Adversary classes. We distinguish four representative adversary types to reflect realistic medical collaboration settings:

(1) External network attacker: a probabilistic polynomial-time (PPT) adversary $\mathscr{A}_{net}$ that fully controls inter-hospital communication (eavesdrop/replay/delay/inject) but has no legitimate registry-bound signing keys.

(2) Malicious (or careless) end users: a patient/doctor adversary $\mathscr{A}_{usr}$ that holds its own legitimate keys and attempts to abuse delegation/re-delegation (e.g., over-delegation, policy escalation attempts, unauthorized re-delegation, or sharing token secrets off-protocol).

(3) Compromised or malicious hospital nodes: a node adversary $\mathscr{A}_{hosp}$ that may deviate from the protocol, selectively drop/alter messages, or leak plaintext and locally held secrets (including K_{AES} when the node is a data holder, and SK_H when compromised).

(4) Ledger-reading consortium participants: a semi-honest adversary $\mathscr{A}_{led}$ with read access to all on-chain state/confirmed transactions (and thus access to authorization metadata and audit logs), but not necessarily to off-chain ciphertext stores unless separately authorized.

System and chain assumptions: We assume deterministic execution with finality in the permissioned chain. The underlying PBFT deployment satisfies the standard safety/liveness condition $f < n/3$, so confirmed transactions are final and tamper-evident within the consortium governance boundary. On-chain requests require ECDSA authorization under registry-bound public keys, so injected messages cannot pass contract checks without valid signatures; replays are mitigated by timestamp validation for delegation requests and by contract state-machine invariants for token enrollment/consumption.

Key and token compromise boundary: Unless stated otherwise, long-term keys SK_P, SK_D, SK_H are uncompromised. If a long-term key is compromised, the adversary inherits the corresponding on-chain authority (e.g., a compromised SK_D can authorize token consumption), so our guarantees degrade to what can be achieved by operational controls (HSM-backed keys, key rotation/revocation, short T_{exp}, bounded $Times$, and auditability). In particular, SST-MedChain cannot cryptographically prevent a fully compromised data-holding hospital from leaking plaintext EMRs that it legitimately decrypts; instead, we aim to minimize exposure (least-privilege and short-lived policies/keys) and provide accountability

through tamper-evident logging. In contrast, compromise of token secrets $\{s_i\}$ alone does not enable unauthorized on-chain consumption: token consumption additionally requires a valid signature under the current token owner's registry-bound signing key (Section 4.1), which is verified on-chain before the state transition. Token secrets are never posted on-chain and are forwarded in re-delegation only as ciphertexts under the delegatee's public key.

Metadata leakage boundary: SST-MedChain records authorization states and audit logs on-chain for accountability and stores target descriptors that resolve to EMR ciphertext in the consortium IPFS. As a result, a ledger-reading adversary $\mathscr{A}_{led}$ can infer metadata such as access timing/frequency, access-scope identifiers, and requester hospital identifiers; if the storage layer is broadly reachable, these target descriptors may additionally enable ciphertext retrieval within the IPFS deployment. We therefore treat such exposure as an explicit system boundary in the threat model. Practical mitigation mechanisms are discussed separately in Section 5.3.

Hospital nodes and off-chain leakage boundary: In the basic model, hospital nodes are honest-but-curious: they follow the protocol to submit user-authorized transactions, but may attempt to infer information from metadata and may observe plaintext during local decryption. We additionally consider the stronger $\mathscr{A}_{hosp}$ model (compromised/malicious hospital nodes). SST-MedChain protects against unauthorized on-chain state changes, replay/double-spending, and in-transit key disclosure in cross-hospital retrieval; however, it cannot prevent a compromised node from leaking plaintext EMR once it legitimately obtains and decrypts it. The design therefore minimizes off-chain exposure by keeping EMR ciphertext in IPFS, discarding K_{AES} after use, using short-lived delegation policies (T_{exp}), and recording authorization/access events on-chain for post-hoc accountability and incident response.

On-chain state model: For the security analysis, we focus on the on-chain state maintained by the DMC. It can be viewed as three categories of data structures, each instantiated by one or more mappings:

- Token Registry: Per-token mappings that maintain the lifecycle status and ownership of $tid = H(w)$, including $Status[tid]$, the owner pointer $DidOf[tid]$, and an optional binding digest $Bind[tid]$ used for target verification.
- Delegation Lists: Mappings indexed by delegation identifier did that store request parameters ($DelegationRequestList$) and verified policies ($VerifiedDelegationList$).
- Lineage and Indices: A delegation lineage pointer $PrevDid[did]$ (for traversing delegation chains) and a root-token index $Roots[did]$ for root revocation (where the root $did = H(M)$).

Other contracts maintain additional state outside the DMC (e.g., $PID \mapsto PK$ registries in SMC and $(PID_P, AccessScope) \mapsto \text{CID}^*$ target bindings in OMC), which we treat as trusted registries in the threat model and omit here for brevity.

We use SHA-256 outputs as 256-bit identifiers. In this paper, we write $tid = H(w)$ and model the DMC state as the mappings $Status[tid] \in \{Unused, Used, Frozen, Revoked\}$, $DidOf[tid]$ (current owner delegation), $Bind[tid]$ (target-verification digest), $DelegationRequestList[did] = \langle c, \mathscr{P}, R, PID_P, PID_D \rangle$, $VerifiedDelegationList[did] = \langle PID_P, PID_D, \mathscr{P} \rangle$, $PrevDid[did]$ (delegation lineage), and $Roots[did]$ (root-token index). At consumption time, the contract additionally records a binding hash to enable verifiable target checks.

5.2. Security analysis

Security goals: We track the following failure events, where $\Pr[\cdot]$ is over protocol randomness and $\mathscr{A}$'s coins:

- E_{replay}: $\exists tid$ such that $Acc(tid)$ occurs twice.
- E_{key}: in a real cross-hospital retrieval, $\mathscr{A}$ recovers the encapsulated session key material K_{AES} from $CT_{K_{AES}}$ without the required secret inputs.
- E_{emr}: $\mathscr{A}$ outputs EMR from CT_{EMR} without authorized decryption.
- E_{forge}: $\mathscr{A}$ forges a valid delegation for some (PID_P, PID_D).
- E_{steal}: $\exists tid$ such that a non-owner consumes tid.
- E_{fwd}: $\mathscr{A}$ recovers any forwarded token secret s_i from a re-delegation ciphertext CT_{fwd} without the delegatee's private key.

Game-based indistinguishability notion for the hospital-side KEM: Separately from the concrete recovery event E_{key}, we define the standard left-right indistinguishability advantage for the token-bound KEM. In the experiment, the adversary chooses two equal-length candidate keys (K_0, K_1), receives a challenge ciphertext encapsulating K_b for a hidden bit $b \leftarrow \{0,1\}$, and outputs a guess b'. We write

$$\mathrm{Adv}^{\text{KEM-IND}}_{\mathscr{A}}(\lambda) := \left| \Pr[b' = b] - \frac{1}{2} \right|. \tag{11}$$

Assumptions: We use the following standard advantages (all as functions of λ): ECDSA existential unforgeability under chosen-message attacks (EUF-CMA), hash preimage/second-preimage resistance, Decisional Diffie–Hellman (DDH), Computational Diffie–Hellman (CDH), HKDF pseudorandom function (PRF) security, and nonce-respecting AES-GCM authenticated-encryption security.

Throughout, we model $H : \{0,1\}^* \to \{0,1\}^\kappa$ as a hash over byte strings, and every structured protocol object (e.g., tuples, $\mathscr{W}$, and the target descriptor CID*) is hashed via an unambiguous canonical byte encoding (in particular, w_i uses SEC1 compressed encoding as required in Section 4.1).

For re-delegation secret forwarding, we use the concrete hybrid encryption instantiation described in Section 4.2 and assume AEAD security of AES-GCM together with CDH/PRF security for the ECDH+HKDF key derivation, so the construction provides standard confidentiality and ciphertext-integrity guarantees for the forwarded secret bundle under the stated composition assumptions. Across the protocol, AES-GCM is used in the nonce-respecting setting: IV_{emr}, IV, and IV_f are sampled freshly and never reused under the same encryption key.

$$\mathrm{Adv}^{\text{Pre}}_{\mathscr{A}}(H, \lambda) : \Pr\left[H(x') = y : x \leftarrow \{0,1\}^\kappa;\ y \leftarrow H(x);\ x' \leftarrow \mathscr{A}(y)\right]. \tag{12}$$

$$\mathrm{Adv}^{\text{DDH}}_{\mathscr{A}}(\mathbb{G}, \lambda) := \left| \Pr\left[\mathscr{A}(g, ag, bg, abg) = 1\right] - \Pr\left[\mathscr{A}(g, ag, bg, cg) = 1\right] \right|, \tag{13}$$

where $a, b, c \leftarrow \mathbb{Z}_q$ uniformly.

$$\mathrm{Adv}^{\text{CDH}}_{\mathscr{A}}(\mathbb{G}, \lambda) := \Pr\left[\mathscr{A}(g, ag, bg) = abg : a, b \leftarrow \mathbb{Z}_q\right]. \tag{14}$$

Delegation authenticity (verification soundness): A delegation authentication request is accepted on-chain only if $\mathrm{Verify}(PK_P, H(M), \sigma_P) = 1$ and the timestamp is valid, which means that any accepted

creation of $DelegationRequestList[did]$ for $did = H(M)$ is attributable to PID_P under the stated key-compromise and registry-integrity assumptions. A delegation verification (enrollment) passes on-chain only if

$$\text{Verify}(PK_D, H(did, N', \mathscr{W}), \sigma_D) = 1 \wedge H(N') = c. \tag{15}$$

Therefore, completing enrollment as D without knowledge of SK_D implies forging an ECDSA signature (or breaking the integrity of the identity registry that binds PID_D to PK_D), while producing a valid N' for the stored commitment implies breaking the preimage resistance of H:

$$\Pr[\mathsf{E}_{forge}] \leq 2 \cdot \text{Adv}_{\mathscr{A}}^{\text{ECDSA}}(\lambda) + \text{Adv}_{\mathscr{A}}^{\text{Pre}}(H, \lambda) + \varepsilon_{smc}. \tag{16}$$

The factor 2 comes from a union bound over the two on-chain signature checks: forging an accepted delegation implies forging either the patient's signature σ_P on $H(M)$ or the doctor's signature σ_D on $H(did, N', \mathscr{W})$, each bounded by $\text{Adv}_{\mathscr{A}}^{\text{ECDSA}}(\lambda)$ under EUF-CMA.

Key indistinguishability via token-bound KEM: In cross-hospital retrieval, the target hospital derives the token-bound shared secret (Equation (3)), the session key (Equation (4)), and then produces the encapsulation packet (Equation (6)) via AEAD (Equation (5)). We analyze the outsider setting in which the input points pass the validation checks of Section 4.1, PK_H is the registry-bound public key of the requesting hospital, $PK_{Com} \neq \mathscr{O}$, and the adversary does not know either SK_H or the token secret s_i.

Theorem 1 (Token-bound KEM Indistinguishability): Under the DDH assumption in $\mathbb{G}$, the PRF security of HKDF, and the nonce-respecting IND-CPA security of AES-GCM, the proposed token-bound KEM satisfies

$$\text{Adv}_{\mathscr{A}}^{\text{KEM-IND}}(\lambda) \leq \text{Adv}_{\mathscr{A}}^{\text{DDH}}(\mathbb{G}, \lambda) + \text{Adv}_{\mathscr{A}}^{\text{PRF}}\big(\text{HKDF}, \lambda\big) + \text{Adv}_{\mathscr{A}}^{\text{IND-CPA}}\big(\text{GCM}, \lambda\big). \tag{17}$$

Proof (Sequence of Games): We use a standard game-hopping argument. Let p_i denote the probability that the adversary guesses the hidden bit correctly in Game i.

Game 0 (Real Left-Right Experiment): This is the real token-bound KEM experiment, so

$$\text{Adv}_{\mathscr{A}}^{\text{KEM-IND}}(\lambda) = \left| p_0 - \frac{1}{2} \right|. \tag{18}$$

Game 1 (DDH Replacement): The simulator replaces the real Diffie–Hellman shared secret point S (Equation (3)) with a random group element $Z \leftarrow \mathbb{G}$. Any distinguisher between Game 1 and Game 0 yields a DDH adversary, hence

$$|p_0 - p_1| \leq \text{Adv}_{\mathscr{A}}^{\text{DDH}}(\mathbb{G}, \lambda). \tag{19}$$

Game 2 (PRF Replacement): Since the input to HKDF is now random and independent of the challenge bit, we replace $K_{session} = \text{HKDF-SHA256}(\text{Encode}(Z))$ with a uniform random string $K \leftarrow \{0,1\}^{\kappa}$. By PRF security of HKDF,

$$|p_1 - p_2| \leq \text{Adv}_{\mathscr{A}}^{\text{PRF}}\big(\text{HKDF}, \lambda\big). \tag{20}$$

Game 3 (Ideal AEAD Challenge): With a truly random session key independent of the challenge bit, we replace the AES-GCM encryption of the selected key K_b with the encryption of a fixed dummy string of the same length. Any distinguisher between Game 3 and Game 2 breaks IND-CPA security of

AES-GCM, so

$$|p_2 - p_3| \leq \mathrm{Adv}^{\mathrm{IND-CPA}}_{\mathscr{A}}\big(\mathrm{GCM}, \lambda\big). \tag{21}$$

In Game 3, the challenge ciphertext is independent of b, hence $p_3 = \frac{1}{2}$ and the adversary's distinguishing advantage is exactly zero. Summing the differences across the games yields Theorem 1.

From indistinguishability to key recovery: Let E_{key} be the concrete event that $\mathscr{A}$ outputs the encapsulated K_{AES} from $CT_{K_{AES}}$. If K_{AES} is sampled uniformly from $\{0,1\}^{\kappa}$, then any adversary that recovers K_{AES} with probability ε gives a left-right distinguisher with advantage at least $\varepsilon - 2^{-\kappa}$. Therefore,

$$\Pr[\mathsf{E}_{key}] \leq \mathrm{Adv}^{\text{KEM-IND}}_{\mathscr{A}}(\lambda) + 2^{-\kappa}. \tag{22}$$

EMR confidentiality: With $CT_{EMR} = (IV_{emr}, C_{emr}, Tag_{emr}) = \text{AES-GCM\_enc}(K_{AES}, IV_{emr}, EMR)$, let E_{emr} denote the event that $\mathscr{A}$ outputs EMR from CT_{EMR}. Then, by a union bound,

$$\Pr[\mathsf{E}_{emr}] \leq \Pr[\mathsf{E}_{key}] + \mathrm{Adv}^{\mathrm{AEAD}}_{\mathscr{A}}\big(\mathrm{GCM}, \lambda\big). \tag{23}$$

Replay/double-spending resistance: Let $Acc(tid)$ denote a successful on-chain authorization for token identifier tid. The contract enforces the invariant

$$Acc(tid) \Rightarrow\ Status[tid] = Unused \wedge\ \big(Status[tid] \leftarrow Used\big), \tag{24}$$

thus, for any fixed tid, E_{replay} can only occur if the smart-contract invariant is violated or finality assumptions fail. Writing ε_{sc} for the probability of such a violation (e.g., contract bug, inconsistent execution, or broken finality),

$$\Pr[\mathsf{E}_{replay}] \leq \varepsilon_{sc}. \tag{25}$$

Unauthorized consumption resistance (owner binding): The above replay argument ensures that a fixed tid cannot be consumed twice, but it does not by itself exclude the case where a *non-owner* attempts to consume tid once after observing the on-chain identifier tid (and, in some deployments, the corresponding w off-chain). We therefore model E_{steal} and rely on the access-control check stated in Section 4.1: the contract binds each tid to a delegation identifier via $DidOf[tid]$ and derives the current owner as $VerifiedDelegationList[DidOf[tid]].PID_D$, allowing consumption only if the transaction is authorized by that PID_D. Under correct contract execution, uncompromised signing keys, and a correct binding from PID to public key in the SMC registry,

$$\Pr[\mathsf{E}_{steal}] \leq \mathrm{Adv}^{\mathrm{ECDSA}}_{\mathscr{A}}(\lambda) + \varepsilon_{smc} + \varepsilon_{sc}, \tag{26}$$

where ε_{smc} captures the probability of a compromised or inconsistent $PID \mapsto PK$ binding in the identity registry.

Least-privilege under re-delegation: On-chain verification enforces the policy monotonicity constraint

$$\forall n:\ \mathscr{P}_{n+1} \subseteq \mathscr{P}_n, \tag{27}$$

and Nested Freezing preserves conservation of spendable tokens through the Unused→Frozen→Unused transitions during re-delegation (Algorithm 2).

Confidentiality of forwarded token secrets: For re-delegation, the delegator forwards token secrets

using $CT_{fwd} = (R_f, IV_f, C_f, Tag_f)$, where the encryption key is derived from the ECDH shared secret S_f (Equation (9)) and the payload is protected by AES-GCM with associated data ad (Equation (7)) (Section 4.2). Let E_{fwd} denote the event that a network/ledger adversary recovers any s_i from CT_{fwd} without $SK_{D_{n+1}}$. Under CDH in $\mathbb{G}$, PRF security of HKDF, and AEAD security of AES-GCM,

$$\Pr[\mathsf{E}_{fwd}] \;\le\; \mathrm{Adv}^{\mathrm{CDH}}_{\mathscr{A}}(\mathbb{G},\lambda) + \mathrm{Adv}^{\mathrm{PRF}}_{\mathscr{A}}\big(\mathrm{HKDF},\lambda\big) + \mathrm{Adv}^{\mathrm{AEAD}}_{\mathscr{A}}\big(\mathrm{GCM},\lambda\big). \tag{28}$$

Binding the associated data ad to $(PID_P, PID_{D_{n+1}}, did_{new}, \mathscr{P}_{n+1}, T_{exp})$ prevents ciphertext re-binding across contexts (the decryptor rejects if ad mismatches).

Policy monotonicity and forwarding-context soundness: Let E_{pol} be the event that the contract accepts a re-delegation with $\mathscr{P}_{n+1} \nsubseteq \mathscr{P}_n$, and let E_{ctx} be the event that a forwarded-secret ciphertext is accepted under a context $(PID'_P, PID'_{D'}, did', \mathscr{P}', T'_{exp}) \neq (PID_P, PID_{D_{n+1}}, did_{new}, \mathscr{P}_{n+1}, T_{exp})$. Since the contract explicitly checks policy inclusion and the delegatee decrypts only under the AEAD-bound associated data, we obtain

$$\Pr[\mathsf{E}_{pol}] \le \varepsilon_{sc}, \tag{29}$$

and

$$\Pr[\mathsf{E}_{ctx}] \le \mathrm{Adv}^{\mathrm{AEAD}}_{\mathscr{A}}\big(\mathrm{GCM},\lambda\big). \tag{30}$$

Thus, unauthorized policy expansion would require a failure of the on-chain state machine, while successful context re-binding of CT_{fwd} would require breaking the integrity of the AEAD layer.

Auditability and non-repudiation: All authentication, enrollment, access, re-delegation, and revocation requests are signed and recorded as confirmed transactions. Under the permissioned-chain assumption and the PBFT safety condition, the ledger provides a tamper-evident audit trail for accountability within the consortium governance boundary, while the DMC state machine maintains a unique, verifiable lifecycle for each token.

Security conclusion: Define the aggregate failure event

$$\mathsf{E}_{sec} = \mathsf{E}_{forge} \vee \mathsf{E}_{key} \vee \mathsf{E}_{emr} \vee \mathsf{E}_{replay} \vee \mathsf{E}_{steal} \vee \mathsf{E}_{fwd} \vee \mathsf{E}_{pol} \vee \mathsf{E}_{ctx}. \tag{31}$$

By a union bound and the inequalities above,

$$\begin{aligned} \Pr[\mathsf{E}_{sec}] \le\; & \Pr[\mathsf{E}_{forge}] + \Pr[\mathsf{E}_{key}] + \Pr[\mathsf{E}_{emr}] + \Pr[\mathsf{E}_{replay}] \\ & + \Pr[\mathsf{E}_{steal}] + \Pr[\mathsf{E}_{fwd}] + \Pr[\mathsf{E}_{pol}] + \Pr[\mathsf{E}_{ctx}]. \end{aligned} \tag{32}$$

Hence, under the stated assumptions on ECDSA, H, DDH in $\mathbb{G}$, HKDF, and AES-GCM, the dominant cryptographic terms are negligible in λ, while replay/double-spending, unauthorized consumption, and policy-expansion abuse are bounded by the system-level failure probabilities $(\varepsilon_{sc}, \varepsilon_{smc})$. Therefore, SST-MedChain supports its delegation-authenticity and confidentiality claims, together with misuse-resistant and auditable authorization, under the stated threat model and operational boundaries.

5.3. System-level limitations and mitigations

The above analysis focuses on cryptographic and on-chain authorization properties under standard assumptions. We next summarize the main system-level limitations and practical mitigations, especially

under stronger adversaries such as $\mathscr{A}_{hosp}$ and $\mathscr{A}_{led}$.

(1) Fully malicious/compromised hospital nodes: If a data-holding hospital is compromised, it may leak plaintext EMRs or K_{AES} after legitimate decryption; no purely cryptographic protocol can prevent this without changing the trust boundary (e.g., end-user decryption, TEEs, or multi-party key release). Mitigations include: (i) storing K_{AES} in a KMS/HSM-backed keystore with strict access logging and key rotation, (ii) enforcing short T_{exp} and small $Times$ to bound exposure, (iii) requiring on-chain authorization and target verification before any key release (already enforced), and (iv) leveraging audit logs for accountability and anomaly-driven incident response.

(2) Delegation-chain abuse and over-delegation: Malicious users may attempt to spread access by re-delegating broadly. SST-MedChain mitigates this by enforcing monotone policy tightening and token conservation on-chain, and by bounding chain length (d_{max}) and per-delegation token count ($Times$). Operationally, consortium policies can further constrain re-delegation (e.g., allow only within whitelisted departments/roles recorded in the registry) and trigger alerts when re-delegation frequency or breadth deviates from expected clinical patterns.

(3) On-chain metadata leakage: Ledger readers can infer timing/frequency and coarse scopes from on-chain logs, and plaintext target descriptors enable ciphertext retrieval if the IPFS layer is reachable. Practical mitigations include: (i) restricting IPFS to a consortium-private swarm/gateway, (ii) logging the minimum necessary fields and using coarse-grained access scopes, and (iii) optionally storing only an opaque handle on-chain and releasing the target descriptor off-chain after authorization, while preserving verifiable target checks by binding $H(\mathrm{CID}^*)$ on-chain.

(4) Registry binding failures (SMC/OMC integrity): SST-MedChain relies on correct bindings from identifiers to public keys and record indices. Compromise or inconsistency in these registries can undermine authorization (captured as ε_{smc} in the analysis). Mitigations include consortium governance controls for registry updates (e.g., multi-signature approvals), frequent audits, and rapid key rotation/revocation procedures.

(5) Non-retractability after disclosure: Like most access-control systems, revocation prevents *future* access and key release but cannot retract data already decrypted before revocation confirmation. To reduce impact, deployments should prefer short-lived policies, minimize the disclosed scope per token, and ensure logs support post-hoc investigation.

5.4. Complexity analysis

We analyze the complexity of Algorithm 1 and Algorithm 2 to characterize efficiency under high concurrency access and multi-level delegation/re-delegation.

Analysis of Algorithm 1 (Delegation and Access): The design uses a space-for-time trade-off. Off-chain one-time token generation costs $O(k)$ (point multiplications/encoding). On-chain, tokens are stored as fixed-size identifiers $tid_i = H(w_i)$ with constant-size metadata, yielding $O(k)$ storage. The critical path is highly efficient on-chain: an $O(1)$ logical lookup and atomic state transition with respect to the policy/attribute space size, excluding consensus/network delays and hashing/encoding work proportional to the returned CID list size. Cross-hospital retrieval adds a constant number of group operations and one AES-GCM encapsulation/decapsulation per request; the chain hashes w_i as bytes and performs no curve arithmetic.

Analysis of Algorithm 2 (Cascading Re-delegation): The complexity of re-delegation depends on the delegation depth d and the subset size m. Generating a re-delegation request off-chain costs $O(m)$, including selecting the subset and encrypting the corresponding token secrets under the delegatee's public key. On-chain, the "Nested Freezing" ensures that permission transfer is a linear state update operation proportional to the subset size. For revocation, the "Source Circuit Breaker" resolves the root delegation identifier by following $PrevDid[\cdot]$ for at most the realized chain depth d, and then iterates over the root-token index $\mathscr{X} = Roots[did]$ (where $did = H(M)$ is the root delegation identifier), yielding work $O(d + |\mathscr{X}|)$ and worst-case work $O(d_{max} + |\mathscr{X}|)$ under the configured depth threshold. In particular, $|\mathscr{X}|$ is linear in the number of tokens issued under the delegation and does not grow with the realized re-delegation depth.

In addition to asymptotic cost, SCB must fit within the platform's per-transaction execution limits; therefore, $Times$ is set to a small value in our implementation (e.g., $Times = 10$), ensuring SCB iterates over at most $|\mathscr{X}| \leq Times$ root tokens per revocation.

6. Experimental evaluation

We evaluate SST-MedChain on a geographically distributed permissioned blockchain and compare it with ABAC [8] and an ABE-based searchable sharing scheme [19]. In addition to throughput and latency under increasing load and high concurrency, we also provide a broader functional comparison with representative related systems and adversarial security experiments to validate key contract-level security properties.

6.1. Experimental setup

We deployed a cross-regional FISCO BCOS permissioned consortium blockchain on Alibaba Cloud to simulate a WAN medical collaboration environment. The detailed parameters of the experimental environment are listed in Table 2. The experimental code is available at [26].

Table 2. Experimental environment parameters.

Parameter	Value/Description
Blockchain Platform	FISCO BCOS v2.11
Node Topology	4 Nodes (Hangzhou, Beijing, Shenzhen, Chengdu)
Compute Instance	Alibaba Cloud ecs.c9i.large (2 vCPU, 4 GiB RAM)
Operating System	Ubuntu 24.04
Network Bandwidth	3 Mbps (Simulating WAN conditions)
Consensus Protocol	PBFT (Practical Byzantine Fault Tolerance)
Stress Test Tool	java-sdk-demo
Crypto Library	Python ecdsa, HKDF-SHA256, SHA-256 (hash), AES-GCM (AES-128)
Curve	SECP256k1

Workload and baseline parameterization: In our experiments, we configure 1000 patients for both SST-MedChain and the ABAC baseline. Each patient is associated with 10 EMR metadata entries anchored on-chain (e.g., indices that resolve to CIDs), reflecting a small but practical record set per patient for stress testing.

In our experiments, the ABAC baseline is intentionally configured with a simplified policy structure (e.g., three attribute categories). In realistic medical workflows, access decisions often require validating a much larger set of contextual attributes (e.g., diagnosis scope, institution, time window, emergency flag, care-team membership). A full-scale ABAC implementation would incur significantly higher on-chain matching and parsing overhead as the number and size of attribute sets grow. Therefore, our simplified ABAC setup is intentionally favorable to the ABAC model, providing a conservative baseline. Even against this optimized baseline, SST-MedChain demonstrates substantial performance gains by replacing repeated policy evaluation on the critical path with lightweight token lookup and atomic state transitions.

For the MedShare baseline [19], we configure 1000 logical chains, with 10 tokens per chain, and a chain data size of 3–7 data records per chain. We use ABAC and MedShare as representative baselines for policy-matching and ABE-based searchable sharing, respectively; we do not claim an exhaustive re-implementation of all recent healthcare blockchain access-control schemes in this experimental section.

All schemes are deployed on the same FISCO BCOS WAN testbed (Table 2) and are driven by the same transaction submission pipeline, where transactions are submitted via java-sdk-demo. To avoid pipeline-induced bias, all schemes use the same client-side concurrency control, request scheduling, and transaction confirmation rule (waiting for a committed receipt). For each target QPS point, we run the stress test for 30 s and report the aggregated throughput and average confirmation latency. For the high-frequency data access stress test, we measure latency as the on-chain transaction confirmation time (from submission to a committed receipt), while excluding low-frequency setup steps (delegation creation/verification and user registration) for all schemes.

Performance metrics: We primarily evaluate two key indicators:

- Transactions Per Second (TPS): $TPS = \frac{N_{tx}}{T}$, where N_{tx} is the number of committed transactions within duration T.
- Average Latency: Mean on-chain confirmation latency from transaction submission ($t_{request}$) to committed receipt return ($t_{response}$):

$$\text{Latency}_{avg} = \frac{\sum_{i=1}^{N_{tx}} (t_{response,i} - t_{request,i})}{N_{tx}} \tag{33}$$

  where $t_{request}$ is the client-side submission time and $t_{response}$ is the time when a committed receipt is returned.

6.2. *Performance analysis of key operations*

We evaluate the latency and throughput of critical operations, focusing on the Data Access path (the high-frequency runtime load).

Efficiency of data access: Figure 4 illustrates the performance of the data access phase under nominal loads ranging from 50 to 300 QPS.

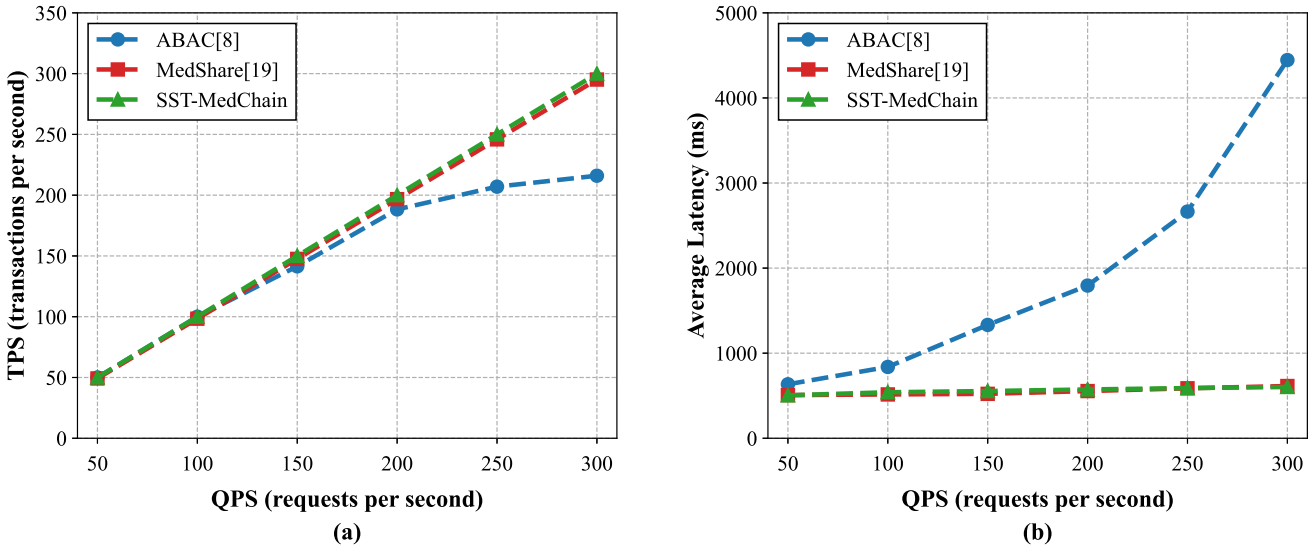

Figure 4. Comparison of **(a)** Throughput and **(b)** Average Latency of different schemes under different QPS.

- TPS: As shown in Figure 4a, the ABAC baseline exhibits saturation at approximately 200 QPS due to the overhead of on-chain policy parsing and attribute matching. In contrast, SST-MedChain scales approximately linearly within the evaluated range, achieving a 38% throughput improvement over ABAC at 300 QPS.
- Average Latency: Figure 4b shows consistently lower on-chain confirmation latency, reducing it by 86% compared to ABAC at 300 QPS, because authorization reduces to token lookup and state update (contract computation) rather than policy tree evaluation.

Discussion on the MedShare baseline: We also compare against MedShare [19], an ABE-based searchable sharing scheme, in the subsequent stress test. Under nominal loads (50–300 QPS), MedShare can exhibit a throughput/latency profile close to SST-MedChain, which is consistent with consensus and network delays dominating end-to-end confirmation time in this WAN setting. We further increase the load to 1000 QPS (Figure 5) to highlight the divergence in throughput and latency under high concurrency; note that we report on-chain confirmation latency and do not decompose MedShare's off-chain cryptographic overhead.

Scalability under High Concurrency: To stress-test the access verification mechanism, we increased the load to 1000 QPS, comparing SST-MedChain with the MedShare scheme [19] (Figure 5).

- Throughput Resilience: In Figure 5a, MedShare [19] saturates around 900 QPS under high concurrency, while SST-MedChain exhibits near-linear scaling within the evaluated range and achieves a 16.5% throughput gain over MedShare at 1000 QPS.
- Latency Profile: In Figure 5b, MedShare exhibits large latency spikes (up to 12,480 ms) under high load, which is consistent with transaction queueing/backlog and delayed confirmation in the testbed; in contrast, SST-MedChain remains comparatively stable (about 2110 ms).

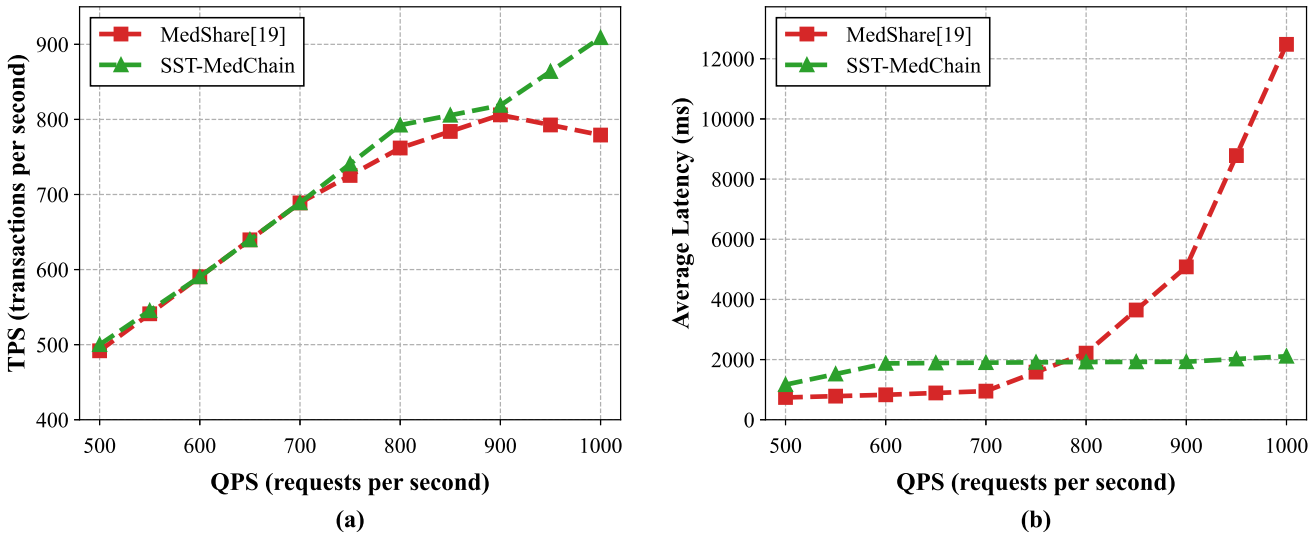

Figure 5. Comparison of **(a)** Throughput and **(b)** Average Latency between SST-MedChain and [19] under high concurrency.

Efficiency of Re-delegation and Revocation: We further evaluated the latency of the cascading re-delegation and revocation operations defined in Algorithm 2 by varying the permission chain length (depth 0 to 10), as shown in Figure 6. Since on-chain updates scale with the number of involved tokens, we fix the number of tokens shared at each re-delegation step as $10 - \text{depth} + 1$ in this experiment. In Figure 6a, depth $= 0$ corresponds to revocation without re-delegation. The average revocation latency (Figure 6a) varies only slightly (550–600 ms) across depths, indicating that consensus dominates over contract execution. Re-delegation latency (Figure 6b) is similarly insensitive to depth, suggesting that Nested Freezing confines on-chain updates to the affected token subset.

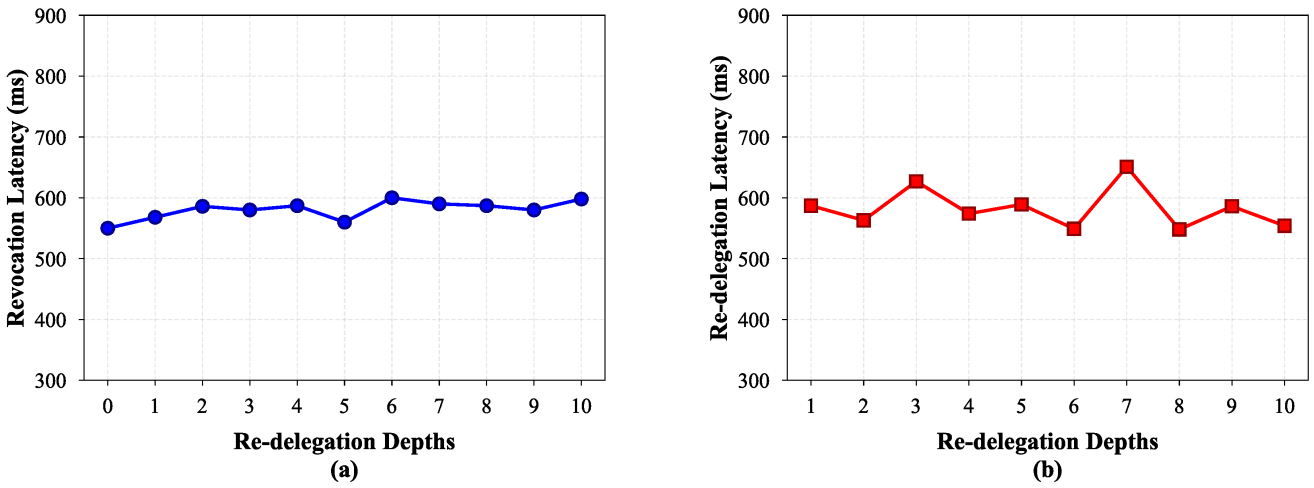

Figure 6. At different delegation depths: **(a)** Time expenditure for the patient to revoke the delegation; **(b)** Time expenditure required to execute re-delegation.

6.3. Functional comparison

To contextualize our contributions, we compare SST-MedChain against recent representative blockchain access-control schemes, encompassing both medical-specific applications and broader generalized data-sharing scenarios (e.g., IoT and edge-assisted environments).

The qualitative ratings in Table 3 are assigned according to the dominant design characteristics explicitly reported in the cited schemes, rather than from a unified re-implementation under our testbed. In particular, "On-chain Overhead" reflects whether the critical authorization path requires heavy on-chain cryptographic/policy evaluation versus lightweight state lookup/update, while "Adaptive Governance" and "Scalability" summarize each scheme's reported support for temporary delegation flexibility and high-concurrency operation.

Table 3. Comparison of access control schemes for medical and generalized data sharing.

Schemes	Access Control Model	On-chain Overhead	Delegation Mechanism	Adaptive Governance	Scalability (High Concurrency)
[19,27,28]	CP-ABE	High	None	Low	Low
[29]	ABAC	High	None	Low	Low
[25]	MA-ABE	High	Interactive	Low	Low
[18]	Edge-Assisted + PRE	Medium	Interactive	Medium	Medium
[22]	Cloud-IoT Smart Contract	Medium	Interactive	Medium	Medium
[24]	6G-enabled Smart Contract	Medium	Interactive	Medium	Medium
[20,30]	State-based RBAC	High	None	Low	Low
[23]	NFT-based Ownership	High	Interactive	Low	Low
SST-MedChain (Ours)	**Token-bound Delegation**	**Low**	**Non-interactive**	**High**	**High**

As summarized in Table 3, many recent studies, including [19, 25, 27–29], favor ABE-based mechanisms to realize fine-grained access control. While such designs offer expressive policy enforcement, they also rely on heavy cryptographic computation. Once access-policy verification or permission updates are pushed onto blockchain-assisted workflows, these schemes tend to incur high on-chain overhead. In high-frequency data sharing scenarios, this design choice can easily lead to throughput bottlenecks and limited agility for adaptive governance.

To relieve blockchain-side computation, another line of work introduces edge computing, IoT devices, or 6G-assisted networking, as exemplified by [18,22,24]. Although these schemes partially offload computation, their delegation handling still commonly depends on interactive coordination across multiple nodes or devices. This interactive logic increases communication complexity and can make latency unstable when network load surges. Meanwhile, traditional smart-contract, RBAC, or NFT-oriented designs such as [20,23,30] provide baseline authorization functionality, but their permission

management remains tightly coupled to heavyweight on-chain state transitions. As a result, they offer only moderate flexibility for temporary authorization and multi-hop delegation.

In contrast, SST-MedChain represents a different design point. By binding permissions to one-time tokens, it realizes non-interactive delegation and converts authorization on the critical path into a lightweight "lookup-and-consume" operation. This design decouples permission lifecycles from heavy stateful interactions, supports adaptive governance, and experimentally shows low latency with a comparatively stable throughput profile under high concurrency, reducing latency by up to 86% and improving throughput by up to 38% over the evaluated baselines.

6.4. Storage and cryptographic overhead analysis

Cryptographic Costs: Figure 7 shows that off-chain cryptographic operations (key encapsulation and verification token derivation (computing N and N')) are millisecond-level; when considered alongside the on-chain confirmation latencies reported in our stress tests, they are not the dominant contributor to overall access latency.

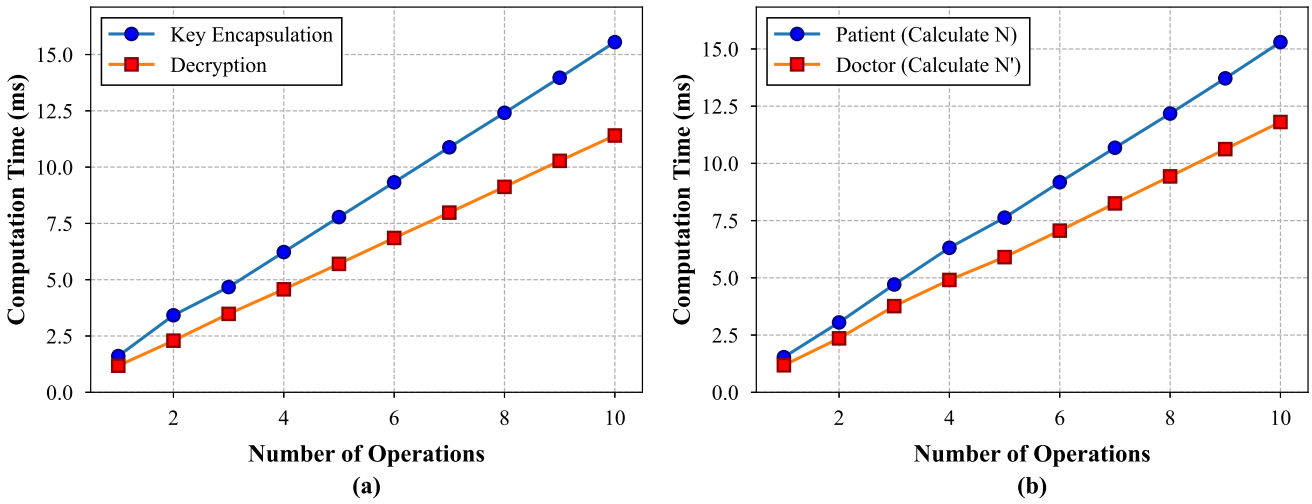

Figure 7. Cryptographic overhead analysis: **(a)** Time overhead of key encapsulation and decryption of encryption keys; **(b)** Time overhead of verification token derivation (computing N and N').

Storage Overhead Analysis: On-chain storage is dominated by per-delegation metadata (c, R, $\mathscr{P}$) and per-token state/binding metadata indexed by $tid = H(w)$ (e.g., bytes32). With $Times = 10$ (*i.e.*, $n = 10$ one-time tokens per delegation), ledger growth is driven by the number of active tokens and constant-size metadata. In our implementation, the node-side ledger database stores 1000 delegation requests in 6.66 MB; after verification, 6.69 MB. Since EMRs remain in IPFS and the chain stores only compact metadata, the storage overhead is manageable in a permissioned setting.

6.5. Adversarial security experiments

This subsection reports adversarial validation beyond efficiency metrics. We evaluate SST-MedChain under adversarial conditions where an attacker lacks valid authorization or operates under an invalid context. The focus here is on results and validation criteria: each case is checked under the criterion "transaction reverts (or returns the expected failure status) + the observed contract-level reason matches

expectation + critical on-chain state is not incorrectly advanced." Across the seven adversarial cases, all attacks are rejected with an attack success rate of 0, and the observed contract-level outcomes match the expected ones.

Each adversarial case is executed against the deployed prototype by submitting intentionally invalid or mismatched transactions through the same contract interface as normal requests. For each case, we record whether the transaction reverts (or returns the expected failure status), whether the reported contract-level reason matches the intended attack category, and whether critical state variables remain unchanged after the attempt.

Specifically, we construct: (1) expired-timestamp replay, where a request is replayed with a timestamp outside the acceptance window to validate time-window anti-replay; (2) double-spending the same token for data access, where the same token is resubmitted after a valid consumption to validate one-time-use semantics; (3) cross-hospital replay of a context-bound signature, where a signature bound to one hospital context is replayed under another hospital identifier to validate context binding; (4) forged-signature re-delegation, where re-delegation is initiated with a non-delegator key to validate delegator authorization; (5) re-delegation with an already used token, where a doctor attempts to create a child delegation from a token that has already been consumed, validating that only currently unused tokens may enter the freeze-and-activate path; (6) policy-disallowed re-delegation, where the parent delegation policy does not grant re-delegation authority in Ops, but the delegated doctor still attempts to create a child delegation; and (7) policy-expansion re-delegation, where a non-subset child policy is submitted to validate on-chain policy-monotonicity checks. Table 4 summarizes the expected outcomes, observed contract-level results, and preserved invariants for these seven cases.

Table 4. Adversarial security experiment results (7 cases; attack success rate: 0).

Attack Scenario	Security Goal	Expected	Observed	Invariant	Attack Succeeds
Expired-timestamp replay	Time-window anti-replay	Out of window	Match	No record created	False
Double-spend consumed token for access	One-time-use authorization	Token unusable	Match	Source Used	False
Cross-hospital signature replay	Context-bound authorization	Signature invalid	Match	State/binding unchanged	False
Forged-signature re-delegation	Delegator authentication	Invalid	Match	No child created	False
Re-delegation with a consumed token	Unused-only transfer eligibility	Invalid	Match	No child created; source Used	False
Policy-disallowed re-delegation	Re-delegation permission binding	Not permitted	Match	No child created	False
Policy-expansion re-delegation	Policy monotonicity	Invalid	Match	No child created	False

Result analysis: These results indicate that SST-MedChain maintains a consistent contract-level defense line against stale-request replay, spent-token reuse, context mismatch, forged re-delegation, re-delegation from an already consumed token, policy-disallowed re-delegation, and policy expansion. First, none of the adversarial cases can incorrectly advance critical on-chain state, which is consistent with the intended state-machine invariants on negative paths. In particular, the "used-token re-delegation" case shows that a token consumed on the access path cannot re-enter the re-delegation pipeline, so the same authorization unit cannot be spent once for access and then re-packaged into a child delegation. Second, the observed contract-level failure reasons are aligned with the attack categories, providing diagnosable and auditable failure semantics. Third, the cases cover the authorization chain more comprehensively, spanning delegation creation $\rightarrow$ token consumption $\rightarrow$ re-delegation eligibility $\rightarrow$ policy-constrained child activation, which provides experimental support for the system claims on abuse resistance and accountability.

7. Conclusion

This paper presents SST-MedChain, a patient-centric framework for tokenized EMR sharing on permissioned blockchains that can be viewed as a tokenized authorization state machine. By combining non-interactive delegation, one-time token authorization, and policy-bounded cascading re-delegation, SST-MedChain decouples high-frequency authorization from heavyweight on-chain policy evaluation and reduces the critical path to a lightweight lookup-and-consume operation. Experiments on FISCO BCOS in a WAN setting show that, on the evaluated data-access confirmation path, SST-MedChain improves throughput by up to 38% and reduces on-chain confirmation latency by up to 86% over ABAC under nominal loads, while achieving 16.5% higher throughput than MedShare with comparatively stable confirmation latency under high concurrency. Together with the security analysis and explicitly stated deployment boundaries, these results support the *secure* and *scalable* objectives within a permissioned consortium setting. The source code is available at [26].

Data availability statement

No supplementary or additional data were generated in this study.

Declaration of generative AI and AI-assisted technologies

During the preparation of this manuscript, the authors used generative AI tools only to improve language and readability. Specifically, the authors used ChatGPT for language polishing only. The authors take full responsibility for the content of the manuscript.

Acknowledgments

Funding: This work was supported by the National Natural Science Foundation of China under Grant 62372121.

Authors' contribution

Conceptualization, Henglong Zhu and Xiaofei Xing; methodology, Henglong Zhu; software, Henglong Zhu; validation, Henglong Zhu, Xiaofei Xing and Guojun Wang; formal analysis, Henglong Zhu; investigation, Henglong Zhu; resources, Xiaofei Xing; data curation, Henglong Zhu; writing—original draft preparation, Henglong Zhu; writing—review and editing, Xiaofei Xing and Guojun Wang; visualization, Henglong Zhu; supervision, Xiaofei Xing; project administration, Xiaofei Xing; funding acquisition, Guojun Wang. All authors have read and agreed to the published version of the manuscript.

Conflicts of interest

The authors declare no competing interests.

References

[1] Zhang R, Xue R, Liu L. Security and privacy on blockchain. *IEEE Trans. Serv. Comput.* 2022, 15(6):3668–3686.

[2] Xu S, Ning J, Li Y, Zhang Y, Xu G, *et al.* A secure EMR sharing system with tamper resistance and expressive access control. *IEEE Trans. Dependable Secure Comput.* 2023, 20(1):53–67.

[3] Xu S, Ning J, Li Y, Zhang Y, Xu G. Blockchain-backed searchable proxy signcryption for cloud Ppersonal Hhealth records. *IEEE Trans. Serv. Comput.* 2023, 16(5):3210–3223.

[4] Prabhakar M, Yenumula K, Katakam A, Bollepally A, Athaluri A. Blockchain based smart healthcare system. In *Proceedings of 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*, Coimbatore, India, June 14–16, 2023, pp. 1480–1484.

[5] Agnal S, Vimal Raj CK, Dinesh Kumar S, Ulagappan GA. Blockchain based electronic medical health records framework. In *Proceedings of 2025 International Conference on Computing and Communication Technologies (ICCCT)*, Chennai, India, 2025, pp. 1–5.

[6] Ou Z, Xing X, He S, Wang G. TDS-NA: blockchain-based trusted data sharing scheme with PKI authentication. *Comput. Commun.* 2024, 218:240–252.

[7] Ullah SS, Oleshchuk V, Pussewalage HSG. A survey on blockchain envisioned attribute based access control for internet of things: overview, comparative analysis, and open research challenges. *Comput. Netw.* 2023, 235:109994.

[8] Batra G. Attribute-based access control. In *Encyclopedia of Cryptography, Security and Privacy*, 2nd ed. Cham: Springer, 2025. pp. 131–133.

[9] de Oliveira MT, Verginadis Y, Reis LH, Psarra E, Patiniotakis I, *et al.* AC-ABAC: attribute-based access control for electronic medical records during acute care. *Expert Syst. Appl.* 2023, 213:119271.

[10] Nweke LO, Yeng P, Wolthusen SD, Yang B. Understanding attribute-based access control for modelling and analysing healthcare professionals' security practices. *Int. J. Adv. Comput. Sci. Appl.* 2020, 11(2):1–8.

[11] Rathee T, Singh P. MedSecureChain: applying blockchain for delegated access in health care. In *New Approaches for Multidimensional Signal Processing*, 1st ed. Singapore: Springer, 2021. pp. 153–163.

[12] Yang Y, Tu Z, Song H, Zhou H. A token-based access control mechanism for the internet of things using blockchain. In *Mobile Internet Security*, 1st ed. Singapore: Springer, 2022. pp. 195–206.

[13] Chen D, Zhang L, Liao Z, Dai H, Zhang N, *et al.* Flexible and fine-grained access control for EHR in blockchain-assisted e-healthcare systems. *IEEE Internet Things J.* 2024, 11(6):10992–11007.

[14] Shahzad A, Chen W, Zhang Y, Kumar R. Zero-trust medical image Ssharing: a secure and decentralized approach using blockchain and the IPFS. *Symmetry* 2025, 17(4):551.

[15] Li F, Liu K, Zhang L, Huang S, Wu Q. EHRChain: a blockchain-based EHR dystem using attribute-based and homomorphic cryptosystem. *IEEE Trans. Serv. Comput.* 2022, 15(5):2755–2765.

[16] Sun Z, Han D, Liu D, Wang X, Chang C, *et al.* A blockchain-based secure storage scheme for medical information. *EURASIP J. Wireless Commun. Networking* 2022, 2022(1):25.

[17] Wu G, Wang S, Ning Z, Yun X, Wang Y. Blockchain enabled privacy preserving access control for data publishing and sharing in the internet of medical things. *IEEE Internet Things J.* 2022, 9(11):8091–8104.

[18] Zhao K, Bao Z, Zhang Y, Lei H. BECAAC: a blockchain and edge computing-assisted access control scheme for medical data sharing. *IEEE Internet Things J.* 2025, 12(16):33376–33394.

[19] Wang M, Guo Y, Zhang C, Wang C, Huang H, *et al.* MedShare: a privacy-preserving medical data sharing system by using blockchain. *IEEE Trans. Serv. Comput.* 2023, 16(1):438–451.

[20] Chu Z, Qiu W, Lei T, He J, Zhang Q. A blockchain-based access control method for large-scale electronic medical records. *Blockchain* 2025, 3(2):12.

[21] Mukta R, Pal S, Hitchens M, Paik Hy, Kanhere SS. Zero trust-driven access control delegation using blockchain. *Blockchain: Res. Appl.* 2026, 7(1):100319.

[22] Alshehri S, Bamasaq O, Alghazzawi D, Jamjoom A. Dynamic secure access control and data sharing through trusted delegation and revocation in a blockchain-enabled cloud-IoT environment. *IEEE Internet Things J.* 2023, 10(5):4239–4256.

[23] Said HE, Al Barghuthi NB, Badi SM, Hashim F, Girija S. PHR-NFT-decentralized blockchain framework with hyperledger and nfts for secure and transparent patient health records. *Appl. Sci.* 2024, 14(22):10744.

[24] Saha S, Das AK, Wazid M, Park Y, Garg S, *et al.* Smart contract-based access control scheme for blockchain assisted 6G-enabled IoT-based big data driven healthcare cyber physical systems. *IEEE Trans. Consum. Electron.* 2024, 70(4):6975–6986.

[25] Duan P, Ma Z, Gao H, Shen Y, Guo Z, *et al.* Multi-authority attribute-based encryption scheme with access delegation for cross blockchain data sharing. *IEEE Trans. Inf. Forensics Secur.* 2025, 20:323–337.

[26] Zhu H, Xing X, Wang G. SST-MedChain. 2026. Available: https://github.com/zhuhenglong-GZHU/SST-MedChain.git (accessed on 6 March 2026).

[27] Li P, Zhou D, Ma H, Lai J. Flexible and secure access control for EHR sharing based on blockchain. *J. Syst. Archit.* 2024, 146:103033.

[28] Ullah A, Ullah Z, Rizvi SS, Gul L, Kwon SJ. Toward blockchain based electronic health record management with fine grained attribute based encryption and decentralized storage mechanisms. *Sci. Rep.* 2025, 15(1):34542.

[29] Kaur J, Rani R, Kalra N. Attribute-based access control scheme for secure storage and sharing of EHRs using blockchain and IPFS. *Cluster Comput.* 2024, 27(2):1047–1061.

[30] Rajasekharan A, Koshy R. EMRChain: electronic medical records management system using blockchain. In *Proceedings of 2024 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*, Pune, India, October 17–19, pp. 1–6.