RESEARCH ARTICLE

INTERNATIONAL RESEARCH JOURNAL OF MULTIDISCIPLINARY TECHNOVATION

Asian Research Association

DOI: 10.54392/irjmt2645

Blockchain-Enabled E-Governance Framework for Secure and Transparent Land Record Management

Aseem Khanna [a, *], Pritpal Singh [b], Prikshat Kumar Angra [a]

[a] Department of Computer Application, Lovely Professional University, Phagwara, Punjab, India.
[b] Mittal School of Business, Lovely Professional University, Phagwara, Punjab, India.
* Corresponding Author Email: aseem.27475@lpu.co.in
DOI: https://doi.org/10.54392/irjmt2645
Received: 05-11-2025; Revised: 09-06-2026; Accepted: 25-06-2026; Published: 04-07-2026

Abstract: Land disputes form a substantial share of pending civil cases in India, and the cost is structural: ownership records sit across five separate offices patwari, tehsildar, sub-registrar, survey, and State Revenue Department that have never shared a single source of truth. Existing blockchain proposals for Indian land records either treat the chain as a future replacement for the statutory record, which is legally premature, or build flat permissioned ledgers that ignore the role hierarchy governing the actual workflow. This paper presents a prototype that addresses this gap by modelling role-aware governance, privacy-preserving document handling, and auditable workflow execution within a Hyperledger Fabric-based land-record framework. The proposed system models six administrative roles as distinct Membership Service Providers on Hyperledger Fabric v2.5.4, encodes mutations, encumbrances, and disputes as first-class chaincode states with role-aware endorsement, and combines ciphertext-policy attribute-based encryption for long-lived role-driven document access with identity-based encryption for short identity-and-time-bound disclosures. This 2 in 1 privacy design increases the level of confidentiality and enables administrators to easily track users' activity at any time as needed. With the workload mix applied using publicly released sub-registrar volume data of two districts of Punjab, the benchmark attained a peak throughput of 612 ± 18 tps and had a mean end-to-end latency of 1.84 s at 400 tps on a cluster of 5 nodes. The observed saturation knee was blamed on CouchDB write contention, as opposed to ordering or endorsement being a problem. The framework is important, as it is proposed as a validation and audit overlay to be used in parallel with the statutory record.

Keywords: Blockchain Governance, Hyperledger Fabric, Indian Land Records, Attribute-Based Encryption, Permissioned Ledger.

1. Introduction

1.1 Land Records in India

In India there is no one office "owned" and has the record of who owns a piece of land. Five do. There is a patwari of the village who maintains register of mutations on daily basis. It is attested by tehsildar on changes. As per the Registration Act, 1908 [1] the deed is to be handled by the sub-registrar. The cadastral map is in the control of survey department. Somewhere above all that there is the consolidated record of rights, which the State Revenue Department maintains, under any Land Revenue Code that the State may have. The Digital India Land Records Modernization Programme (DILRMP) [2] covers the digitization of various segments across different States at different stages with schemas designed with no thought for seamless interoperability across State boundaries. In Punjab and Uttar Pradesh it is called as Bhulekh, in Maharashtra considering the name as Mahabhulekh and in Telangana it is e-Dhara and in Gujarat it is called as Dharani. Each developed their own ‘tradition' or ‘etiquette'.

What a permissioned blockchain offers, that the current system doesn't, is a single shared ledger that all the participating offices can read at the same time, with cryptographic proof of who wrote what and when [3, 4]. That isn't, in itself, a legal claim. The question of what counts as the record of rights is a matter of statute, and a chain doesn't change statute. It also doesn't replace offline identity and consent verification, which still need to happen in person. Inside those limits, the chain layer can do three things the current setup can't. It can carry the audit trail. It can move state changes between offices without manual reconciliation. And it can give the citizen a status they can actually see for any open transaction. Most existing blockchain proposals for Indian land records have either treated the chain as a future replacement for the official record (a position that runs ahead of the law) or built generic blockchain designs without modelling the specific roles that govern the

actual workflow: patwari, tehsildar, sub-registrar, registrar, court [5 - 7]. The result is prototypes that show that a blockchain can hold land data, but not prototypes that show how a blockchain fits inside the administrative process as it currently operates. The paper takes the second position. The blockchain is treated as a supplement to the statutory record, with no claim to displacing it.

The prototype here implements six administrative roles as distinct Membership Service Providers. Each MSP has its own certificate authority, and every official's certificate carries a role attribute. Mutations, encumbrances, and disputes are first-class chaincode states with role-aware endorsement policies, so a sub-registrar's signature is required to commit a mutation regardless of which client submitted it. Document content lives off-chain in an encrypted IPFS store [7, 8], and the on-chain ledger holds only structural data, hashed identifiers, and pointers to ciphertexts. Document confidentiality uses CP-ABE for long-lived role-driven access and IBE for short identity-and-time-bound disclosures.

We evaluate the prototype under a synthetic workload derived from publicly available sub-registrar daily-volume data from two Punjab districts. Reported numbers cover peak throughput and latency, the saturation point of the prototype, the subsystem responsible for that saturation, and how the numbers degrade when wide-area latency is introduced through tc-netem. The point of measuring like this is not to set a record but to understand where the system breaks first.

The paper makes four narrow contributions, each verifiable from the prototype:

1. Role-aware governance through six MSPs. The administrative roles that govern Indian land registration are encoded as distinct Membership Service Providers with role-aware endorsement policies, not as a single permissioned organisation. The chaincode rejects any mutation that is not endorsed by both a State Revenue peer and a Sub-Registrar peer with the role registrar attribute, regardless of the originating client.
2. Combined CP-ABE and IBE for document confidentiality. Long-lived role-driven document access uses ciphertext-policy attribute-based encryption; short identity-and-time-bound disclosures use identity-based encryption. To our knowledge this is the first prototype that combines both primitives in the Indian sub-registrar context, and the first to handle revocation through attribute versioning recorded as on-chain events.
3. First-class encumbrance and dispute lifecycles. Encumbrance and dispute states are full chaincode states with court-MSP-gated transitions, rather than passive flags. Resolving a dispute requires endorsement by a Court MSP peer; releasing an encumbrance requires endorsement by the secured creditor's representative.
4. Realistic workload evaluation. Performance is reported against a workload mix derived from publicly available sub-registrar volume distributions, rather than a uniform synthetic load. We trace the saturation knee of the prototype to a specific subsystem (CouchDB write contention) and measure the effect of injected wide-area latency.

We make no cryptographic novelty claim. CP-ABE [9] and IBE [10] are well-established primitives, and Hyperledger Fabric's Raft ordering is part of the released distribution. The contribution is in the integration of these elements around a specific administrative workflow, with reproducible evaluation.

None of these four points is novel by itself. Multi-MSP permissioned ledgers exist. CP-ABE and IBE are textbook primitives. Ideally encumbrance and dispute states could be added to any chaincode. Caliper Benchmarks are common place. Well done to the whole team for getting all four in one prototype, tested on a workload developed from actual Punjab sub-registrar volumes, and making it clear that not all four of these can be achieved at this point without legislation.

2. Literature Review

2.1 International Blockchain Land Records

First blockchain land-registry projects have been public chain based. In 2015, Honduras made an announcement about it. Since 2016 [11] the state of Georgia has begun to use the hash of property to be made available on the Bitcoin blockchain with the assistance of Bitfury. Instead, Sweden's Lantmäteriet launched the Swedish smart-contract pilot, in a permissioned ledger. There were two takeaways from those experiments. First, it's easier to make land records on a chain time stamped. The political issue is more difficult: who will be the controllers of the writers? Second: oftentimes, it is not the registry that is the problem in property conveyancing. It is the system of banks, brokers and notaries which surrounds it. A blockchain is able to coordinate that chain. It can't shrink it.

Later projects shifted to permissioned ledgers for the most part because the anchoring onto the public ledger by itself did not alter legal record status. In fact, Estonia is currently doing pretty much as much as can be accomplished without running the full blockchain: with cryptographic hashing you get a similar audit guarantee. The Lantmäteriet study itself eventually concluded that the registry isn't really the issue. The workflow around it

is. That conclusion lines up almost exactly with what the Indian situation looks like.

2.2 Indian Blockchain Land-Record Work

Work in the Indian context has moved through three phases. The first, around 2018-2019, consisted of conceptual proposals papers describing the benefits of moving the land registry to a blockchain without specifying the chain technology, the roles, or the workflow. The second phase 2020-2022, produced functional prototypes; we discuss five anchor studies below. The third phase, ongoing, is concerned with integration with state portals and statutory recognition, but few peer-reviewed results are yet available.

The closest India-focused studies, summarized in Table 1, fall into four groups: public-chain pilots that anchor hashes without modelling roles or document confidentiality [6] blockchain-with-IPFS schemes that store property documents off-chain without modelling the administrative role hierarchy [12] single-MSP Hyperledger Fabric prototypes restricted to registration and relying on transport-layer security alone [13] and blockchain smart-contract designs that define the registration workflow at the contract level, without a deployed-network benchmark [5]. NITI Aayog [14] is a strategy paper rather than a system. Each of these is useful as a piece of the puzzle, but none combines role-aware multi-MSP governance, dual-mode document confidentiality, encumbrance and dispute lifecycles, and workload-realistic evaluation in a single prototype.

2.3 Comparison with the Present Work

Table 1 places the studies above side by side with the present work along six dimensions: chain type, governance model, role granularity, privacy mechanism, workflow scope, and evaluation. The present work differs from each in at least one substantive way; from the closest of them, [13], it differs in role granularity, privacy mechanism, and workflow scope at the same time.

The four contributions stated in Section 1.4 follow directly from this comparison. The novelty is integrative. Governance separation across six MSPs, two-mode document confidentiality through CP-ABE and IBE, full-state encumbrance and dispute lifecycles with court-MSP-gated transitions, and evaluation against a workload mix that reflects observed daily volumes in two Punjab districts: none of the prior India-focused studies above combines all four.

3. Methodology and System Architecture

3.1 Design Goals

Five goals shape the architecture, in rough priority order. The first is tamper-evidence with attribution. Every state change must be recorded with the cryptographic identity of the office that produced it, in a way that no participating organization can rewrite alone. The second is around authorisation to make this work, and it must happen at a role level; patwari, registrar, court, etc, and not at the level of the organisation. Otherwise the policy doesn't work when people change desks in an office.

Table 1. Comparison with the closest India-focused blockchain land-record studies.

Study	Chain	Governance	Privacy	Workflow scope	Evaluation
Thakur *et al.* [6] (AP / Telangana)	BSV public	Single state-org control	None on document layer	Registration only	Throughput cited from BSV literature
Shinde et al. [12]	Blockchain + IPFS	Generic buyer / seller / authority model; no role hierarchy	Documents off-chain in IPFS; pointers on-chain	Registration and document storage	Survey + proposed design; no performance benchmark
Krishnapriya & Sarath [13] (Kerala)	Hyperledger Fabric	Single MSP	TLS only	Registration	Conceptual prototype
Sahai & Pandey [5]	Blockchain (smart contracts)	Smart-contract logic for buyer / seller / official	None	Registration	Conceptual design; not implemented or benchmarked
NITI Aayog [14]	N/A	Conceptual	Conceptual	Conceptual	N/A
This work	Hyperledger Fabric v2.5.4	Six MSPs, role-aware endorsement	CP-ABE + IBE + searchable enc.	Registration, mutation, encumbrance, dispute, appeal	Caliper, India-derived workload

Data minimization then. The ledger never lists out any personal data like the Adhaar number, PAN number, name or address. The hashes and the pointer is stored on the on-chain layer, the documents is stored off-chain and is encrypted. Then operational realism comes next and many people tend to overlook this when they pass things by in theory.

The system must be based on equipment a state revenue department could purchase; it must have enough throughput to handle the daily number of transactions for a typical district, not one that is only seen on a research cluster and can never be reproduced. Last is the Reproducibility - it's mostly a discipline thing. All the parameters which will impact performance are recorded and the prototype's configuration is maintained in version controlled files so that anyone doing this again can achieve the same numbers without guessing.

For emphasis, three things aren't on the list. We didn't try to get Sybil-resistance through public consensus. No content of the documents was put on chain. We didn't make the prototype to a schema for a specific state's port.

3.2 Network Topology

The prototype consists of five operational organizations with a logical Court MSP that is not part of the day to day write path. Figure 1 gives the overall picture of how these pieces fit together, from the client applications at the top down to the off-chain key authority and the IPFS store. There are 5 operational departments & these are State Revenue Department (SRD), District Sub-Registrar (DSR), Tehsil/Patwari office (TPO), Survey Department (SVD) and an Audit observer (AUD). Each maintain and operate a peer node and an orderer node, each have their own CA hierarchy and each have their own MSP defined on the channel. The Court MSP is a little more different – it gives identities for endorsements on the court-side, if a dispute/order should be recorded, but does not maintain a peer continuously online. That reflects what courts are really: not in practice sitting on a permanent connection to the registry.

Two channel set-up is used. Land-channel handles the parcel state and mutation history. Ops-channel carries audit logs, configuration changes, and key-rotation events. Splitting them like this lets us put tighter access controls on operational data without dragging the main mutation path down.

3.3 Ordering Service Configuration

The ordering service is implemented as an etcdraft cluster of five orderer nodes, one per operational organisation, with consensus tolerating the failure of two nodes (n = 5, f = 2, quorum = 3) [3, 15]. Raft itself derives from earlier work on Paxos [16] and is the crash-fault-tolerant alternative to BFT consensus protocols such as PBFT [17] used in some other permissioned ledgers. Configuration values are reproduced below from the prototype's configtx.yaml so that the experiments in Section 6 can be repeated without reconstructing them from prose. TLS is mutual on every gRPC link, with channel-scoped CA hierarchies. What's above is the configuration the prototype actually uses, not an idealised version of it. For a wider-area deployment, BatchTimeout and TickInterval would have to be raised to amortise round-trip cost; Section 6.6 has the numbers.

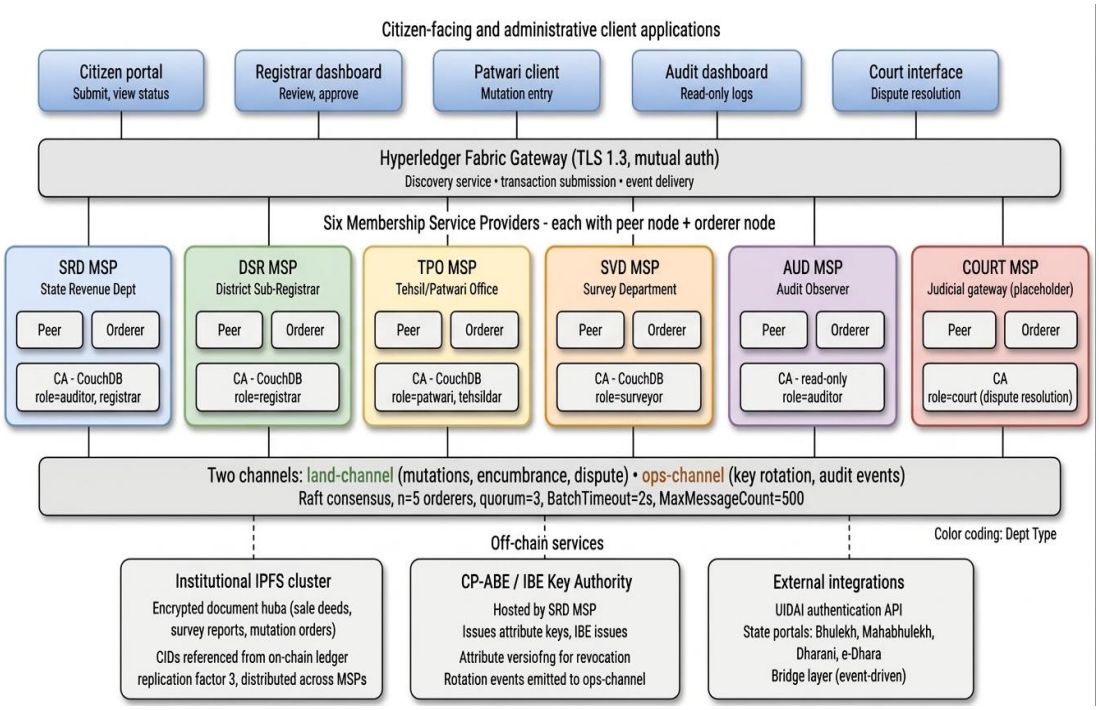

Figure 1. System architecture of five-MSP permissioned blockchain for Indian land record

Orderer:

OrdererType: etcdraft

BatchTimeout: 2s

BatchSize:

MaxMessageCount: 500

AbsoluteMaxBytes: 10 MB

PreferredMaxBytes: 2 MB

EtcdRaft:

Options:

TickInterval: 500ms

ElectionTick: 10

HeartbeatTick: 1

MaxInflightBlocks: 5

SnapshotIntervalSize: 16 MB

Capabilities:

V2_0: true

3.4 Administrative Workflow as a State Machine

This section maps the actual sub-registrar workflow onto chaincode states. Verification, encumbrance check, registrar approval, the mutation itself, dispute, appeal — all of it becomes either a state or a transition. Each state is a value of statusEnum on the parcel record. Each transition is a chaincode function with explicit endorsement and role requirements attached to it. Figure 2 shows the state machine. Table 2 lays those transitions out as a matrix, with the endorsement rule and the exception handling for each one. Table 4 (in §5.2) lists the chaincode functions that go with it.

A mutation begins with an applicant submitting a preliminary deed and documents on the citizen platform. The Parcels record is placed in DRAFT mode. Once submitted to the Patwari it becomes SUBMITTED. Next the patwari refers to the cadastral map to verify surveyNo, confirms parties and finally records ground-level dispute, if any. There are two possibilities from here on. The patwari either rejects it, the record goes to REJECTED and patwari logs "reason" as chaincode event in REJECTED, or the patwari takes it to the VERIFIED_PATWARI. The chaincode will then automatically run query encumbrance ledger. If a charge is outstanding, the parcel is transferred to the LR of ENCUMBRANCE_PENDING and will remain there until the secured creditor provides a ReleaseEncumbrance on the registrar's LR. If not it moves on to REGISTRAR_REVIEW.

The sub-registrar at REGISTRAR_REVIEW reviews the deed to look for any discrepancies between the deed and the Registration Act requirements which include stamp duty, presence of witnesses, checking of identity etc. Approval interaction causes the chaincode to update new_ownerHash, add a TX (transaction) record and then publish a Mutation event to be consumed by the citizen portal.

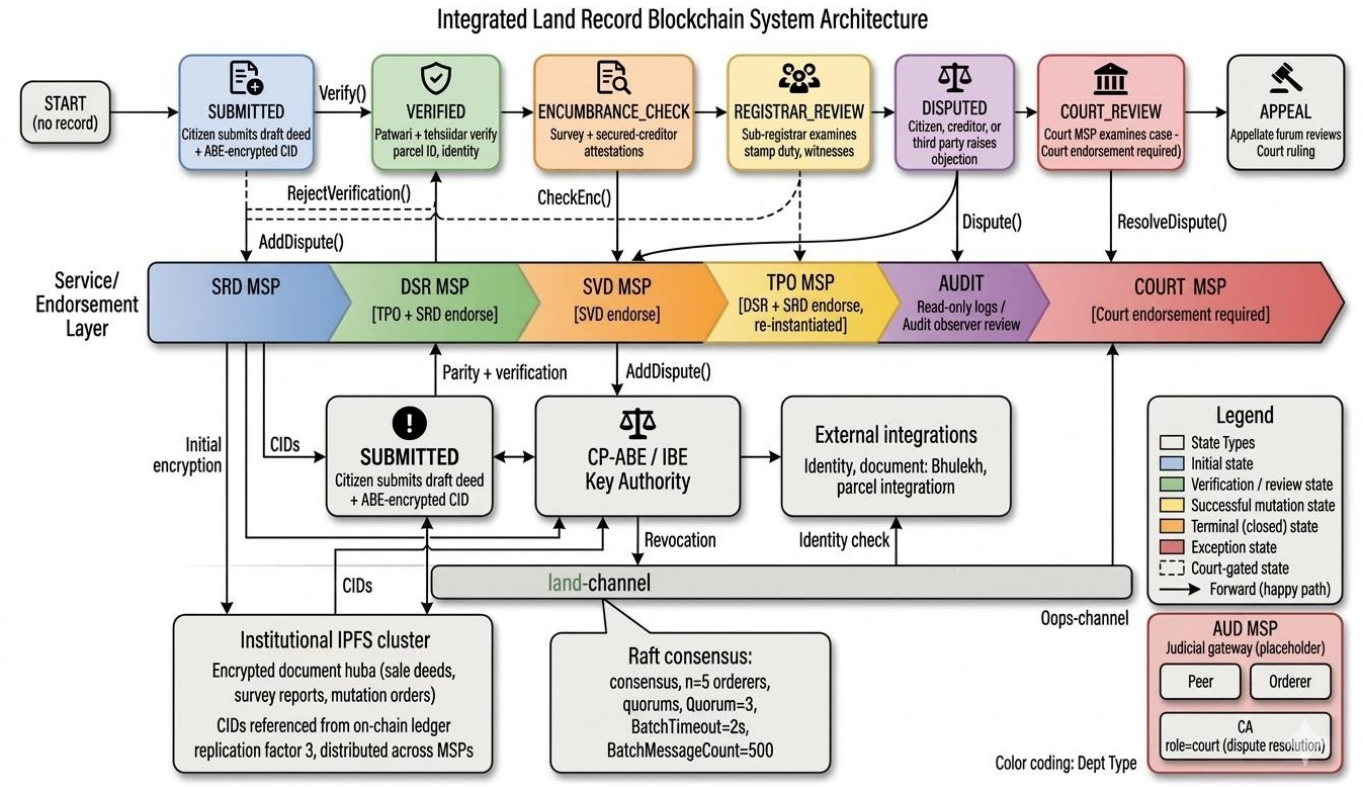

Figure 2. Land Mutation Workflow Implemented as a Chaincode State Machine

Table 2. State-Transition Matrix for the Proposed Blockchain-Enabled Land Mutation Workflow

Current State	Trigger Function	Input	Endorsement Requirement	Next State	Exception Handling
DRAFT	SubmitParcel	ParcelID, DocCID	Citizen + Patwari	SUBMITTED	Invalid data → REJECTED
SUBMITTED	VerifyParcel	SurveyNo, IdentityProof	Patwari MSP	VERIFIED_PATWARI	Mismatch → REJECTED
VERIFIED_PATWARI	CheckEncumbrance	ParcelID	Auto (chaincode query)	ENCUMBRANCE_PENDING / REGISTRAR_REVIEW	Active encumbrance → HOLD
ENCUMBRANCE_PENDING	ReleaseEncumbrance	EncID	Registrar MSP	REGISTRAR_REVIEW	Invalid release → STALL
REGISTRAR_REVIEW	TransferOwnership	OwnerHas, MutationCID	AND(SRD, DSR)	APPROVED	Missing endorsement → FAIL
APPROVED	CommitTransaction	RW-set validation	Fabric validation	MUTATION_RECORDED	MVCC conflict → RETRY
MUTATION_RECORDED	CloseMutation	Time elapsed	System	CLOSED	Dispute → DISPUTED
ANY STATE	AddDispute	CaseID	Court MSP	DISPUTED	Invalid case → IGNORE
DISPUTED	ResolveDispute	CourtOrderCID	Court MSP	REGISTRAR_REVIEW	No court approval → BLOCK

The record is 'blended' and remains in the statutory waiting period until such time as there is no objection to it changing to 'CLOSED'. AddDispute will allow disputes to be entered from the time of SUBMITTED through any state. The parcel is sent to DISPUTED and can be reclaimed in the workflow only via ResolveDispute (which has to be endorsed by a Court MSP). Appeals are treated as a special type of dispute, with caseID as part of appeal being bound to forum of appeal.

Algorithm 1. End-to-End Workflow for Land-Record Mutation

This paper has only one algorithm (Algorithm 1) remained. The detailed pseudocode from prior versions have been summarized in Figure 2 (state-machine model in §3.4) and Table 4 (chaincode API surface in §5.2.2). We maintain Algorithm 1 as this is the only artifact which places all the cross-layer interaction in one place, including the citizen portal, patwari verification, registrar and court action, ABE/IBE services, ordered off-chain storage using IPFS, and the ordering/commit pipeline. Only the interface table – or the state diagram – can capture that integrated flow.

Inputs: applicant a, parcel p, deed draft d, witnesses W

Outputs: committed mutation transaction **or** rejection reason

1. Encrypt + Store Off-Chain

1.1 Define ABE access rule:

pol ← (role = registrar ∧ district = d.district) ∨ (role = audit)

1.2 Compute ciphertext E ← ABE_Encrypt(d, pol)

1.3 Upload E to IPFS and obtain content id c

2. Initialize Workflow on-Chain

2.1 Require(role = citizen)

2.2 Call RegisterParcel(p, c) on land-record-cc

2.3 Set workflow state statusEnum ← DRAFT (or initial submitted state per design)

3. Patwari Validation (Parcel + Identity)

3.1 Registrar issues short-lived IBE token t for controlled disclosure

3.2 Patwari checks cadastral consistency and identity proofs via t

3.3 If checks pass:

Invoke VerifyParcel(p) and set state SUBMITTED → VERIFIED_PATWARI

3.4 Else:

Set state → REJECTED and emit event Reject(reason)

4. Encumbrance Gate

4.1 Query encumbrance index ENC(p)

4.2 If ENC(p) = active:

set state → ENCUMBRANCE_PENDING

block progression until ReleaseEncumbrance(p) is committed by secured creditor via registrar path

5. Registrar-Side Statutory Processing

5.1 Sub-registrar verifies stamp duty, witnesses *W*, and statutory checklist

5.2 Endorse TransferOwnership(p, …) under role = registrar

5.3 Enforce endorsement policy:

AND (StateRevenue.peer, SubRegistrar.peer)

6. Order + Commit

6.1 Orderer batches transaction into a block

6.2 Committing peer validates: endorsements, MSP membership, CRL status, RW-set versions

6.3 If valid: apply state update and emit Mutation event on land-channel

Else: mark transaction invalid and record failure cause per Fabric semantics

7. Closure and Exception Routing

7.1 If waiting period elapses with no objection: set state → CLOSED

7.2 If dispute arises after submission: set state → DISPUTED

7.3 Only court-endorsed ResolveDispute(caseID, …) may return case to the normal path

7.4 Model appeal as dispute subtype with caseID bound to appellate forum

4. Threat Model and Security Analysis

4.1 Adversary Classes and Assumptions

The threat model is organized by capability rather than by attacker name. We consider six adversary classes:

1. Compromised endorsing peer. An adversary controls one peer in some MSP and attempts to endorse fabricated read-write sets.
2. Compromised orderer. An adversary controls one orderer and attempts to omit, reorder, or delay transactions.
3. Compromised registrar workstation. An adversary controls a registrar's client identity (stolen smartcard, key extraction) and attempts to submit unauthorised mutations. Behavioural-analysis approaches to detecting this kind of compromise have been studied in the wider insider-threat literature [18].
4. Malicious external client. An adversary submits invalid, replayed, or grammatically malformed proposals.
5. Network-level adversary. An adversary observes or interferes with traffic between nodes; we assume mutual TLS but not perfect link integrity.
6. Key-management compromise. An adversary obtains an attribute key, an IBE token, or a CA private key.

The model is based on two assumptions. The first, is the root CA of State Revenue Department is trusted – someone breaches that root and can generate arbitrary identity – there's nothing we can do about it other than have hardware security modules. They are used in the prototype, but are not formally analysed here. The second is that the channel for CERT-In incident reporting is available to deal with incidents beyond the CERT chain of command.

4.2 Indicators and Decision Rules are Presented

We list below for each adversary class the measurable signal we use to detect it; the deterministic decision rule used at the endorsing/committing peer; and the Fabric construct that stores the damage. The mapping is summarized in table 3.

4.3 Enforcement via MSP, Endorsement Policies, and Revocation Controls

The Membership Service Provider is in place to keep identity level threats contained. Each of the certificate has a role attribute (registrar, patwari, surveyor, audit, citizen). The policies of endorsement to Transfer Ownership are written against these roles, not the bare names of the Organizations, such that even in the State Revenue MSP, only the certificates with role=registrar will be able to create an endorsement to a proposal of Transfer Ownership. Their mutation transactions need to have AND ('StateRevenueMSP.peer', 'SubRegistrarMSP.peer') and role registrar in at least one of the signatures; and their query transactions need any single peer.

We will be using Fabric's normal CRL mechanism for revocation. The orderer broadcasts a resulting config block which the channel administrator sends a configuration change to use a new CRL in place of the one from the MSP that was suspended, and the CA publishes a new CRL. Starting from this block, any proposal which contains a signing certificate contained in the CRL will be rejected and the identity cannot obtain endorsements any more anymore; the error class will be BAD_CREDS.

Table 3. Integrate threat to measurable signal enforcement mapping.

Threat	Measurable signal	Decision rule	Fabric enforcement
Compromised peer forging endorsement	Signature verification fails against MSP root or intermediate	Reject as BAD_PROPOSAL_RESPONSE	MSP, BCCSP signature check at committer
Replay of an earlier valid tx	Duplicate txID seen on channel	Reject as DUPLICATE_TXID	Committer-side dedup against state DB
Stale read by malicious endorser	Read-set version ≠ current key version	Reject as MVCC_READ_CONFLICT	Validation phase, deterministic
Revoked official attempts mutation	Cert serial appears in MSP revocation_list	Reject as BAD_CREDS	CRL pushed via channel config update
Orderer censorship attempt	Leader fails to replicate within ElectionTick	Trigger leader re-election	Raft (etcdraft) consensus
Sybil registration of citizen	Same Aadhaar hash bound to multiple identities	Chaincode invariant: one active identity per hashed Aadhaar	Application-layer check in RegisterCitizen
Unauthorised read of confidential record	ABE policy not satisfied / IBE token expired	Decryption fails; access logged	Off-chain encryption + on-chain access log

Quarantining a suspect peer (without permanent revocation) is different, however: We simply disable the peer's identity via a less invasive channel update, de-anchor-peers its anchor-peer entry and the peer is no longer part of the gossip dissemination while the investigation goes on.

An example helps. Assume a sub-registrar has lost his/her smartcard on day t. As part of the CRL update, the DSR CA publishes a CRL update, which contains the certificate serial corresponding to t+δ1. When the channel administrator submits channel., an NPE occurs and the thread below appears. Update at t+δ2 (somewhere between δ1 and δ1+30 min) Orderer takes a snapshot of the configuration block at t+δ3. Starting at t+δ3 each endorsement with such a certificate is rejected by the endorsing peer with BAD_CREDS and the rejection is recorded as a security event on ops-channel. Thus the last "window of opportunity" during which the lost certificate can cause any harm is within time interval δ3 which in this prototype is normally less than 1 hour.

4.4 Threats not covered

Two classes of threat are explicitly not addressed by the chain layer. Coercion of an authorized official, where a registrar is forced or bribed to sign a fraudulent mutation, produces a valid blockchain entry. The chain provides audit and non-repudiation, but it cannot detect the absence of free consent; behavioral-analytics layers above the ledger have been proposed for exactly this gap [18]. Off-line falsification of physical evidence (deeds, witness affidavits) before they enter the system is also out of scope. Both belong to the surrounding administrative process, not to the chain. Denial-of-service attacks at the network layer are partly mitigated by Raft's tolerance to f = 2 failures and by gateway rate limits, but a sustained attack on a majority of orderers would stall the channel until quorum is restored; complementary network-layer defenses for blockchain SDN and resource-constrained settings are surveyed in [4, 19].

5. System Implementation

5.1 Software Stack and Runtime

The prototype runs Hyperledger Fabric v2.5.4 on a five-machine cluster. Each machine is an Ubuntu 22.04 LTS VM with 8 vCPUs, 16 GB RAM, and an NVMe-backed disk, all on an isolated 1 Gbps switch. Peer nodes use CouchDB 3.1 as the state database, with indexes pre-built on parcelID and ownerHash. The chaincode is in Go, using the Fabric Contract API v2.5. The registrar, patwari, citizen, and audit dashboards are React 18 against the Fabric Gateway. ABE and IBE both use Charm-Crypto v0.50. Off-chain document storage is an IPFS 0.21 cluster with three replicas. Caliper v0.5.0 drives benchmarks, and every component's configuration is in version control.

5.2 Chaincode Design

5.2.1 State Key Schema

The world state uses composite keys, which makes range queries cheap and means auxiliary indexes (owner → parcels, for instance) don't need a separate database. We've kept the schema deliberately narrow. Each key holds only what's needed to look the

record up, and the values store nothing but structural fields and pointers.

PARCEL~<state>~<district>~<tehsil>~<surveyNo> -> Parcel JSON

OWNER~<sha256(aadhaar|salt)>~<parcelID> -> empty value (index)

TX~<parcelID>~<seq> -> mutation pointer

ENC~<parcelID>~<encID> -> Encumbrance JSON

DISP~<parcelID>~<caseID> -> Dispute JSON

DOC~<parcelID>~<docHash> -> {cid, encScheme, policy}

The Parcel JSON itself stores only structural and pointer fields: parcelID, ownerHash, area, geoCentroid, statusEnum, lastTxID, docBundleCID. Aadhaar number, PAN, name, address — none of those go on-chain in plaintext. Identifiers exist on-chain only in their hashed-with-channel-salt form. The actual document content lives off-chain, referenced through its content-addressed CID.

5.2.2 Function Interfaces

The contract is implemented in Go against the Fabric Contract API v2.5. Every public function follows the same internal pattern: extract caller identity, check role and organisation, validate input shape, read current state, apply the transition, write new state, emit an event. The signature of the access-control helper is reproduced below; remaining functions invoke it as their first executable line.

Func requireRole(ctx contractapi. Transaction Context Interface,
allowed [] string) error {

cid := ctx.GetClientIdentity()

Role, found, err: = cid.GetAttributeValue ("role")

If err! = nil ||! Found {return err Unauthenticated}

For _, r: = range allowed {if r == role {return nil}}

return errForbidden

}

Table 4 lists every chaincode function with its input, output, endorsement requirement, and required role attribute. The table is the implementation-oriented replacement for the four pseudocode listings that appeared in the earlier draft.

5.2.3 Read Write Set, Certificate Validation, Signature Checks

The default Fabric flow behaviour is read-write set. Since the paper is self-contained [3, 20] we're describing it here only to this end. In the simulation, the endorsing peer executes the function on the state DB, and records all the GetState and PutState as respectively the read set and write set of the state DB. Each read set includes the version (block-num, tx-index) of all the keys it has read. When the validation time retrieves the version that the committer previously set in the read set, the set version of his current read set must match the version of the ledger at this moment. If it does not, then the transaction is marked MVCC_READ_CONFLICT and the write set is discarded.

Table 4. Chaincode function interface table.

Function	Input	Endorsement	Required role
RegisterParcel	ParcelMeta, docBundleCID	AND(StateRevenue.peer, SubRegistrar.peer)	patwari + registrar
TransferOwnership	parcelID, newOwnerHash, mutationCID, sigBundle	AND(StateRevenue.peer, SubRegistrar.peer)	registrar
AddEncumbrance	parcelID, encType, lender, amount, expiry	AND(SubRegistrar.peer, StateRevenue.peer)	registrar
ReleaseEncumbrance	parcelID, encID, releaseCID	AND(SubRegistrar.peer, StateRevenue.peer)	registrar
AddDispute	parcelID, caseID, courtCID, courtSig	AND(SubRegistrar.peer, Court.peer)	court
ResolveDispute	parcelID, caseID, orderCID	AND(SubRegistrar.peer, Court.peer)	court
QueryParcel / QueryHistory	parcelID	any single peer	patwari, registrar, audit, owner
GrantTimedAccess (IBE)	parcelID, viewerID, ttl	SubRegistrar.peer	registrar
RegisterCitizen	aadhaarHash, attributeProofs	SubRegistrar.peer	registrar

The two mutations cannot happen at the same time on the same parcel, e.g, even if each mutation in a parcel by itself is well formed. That is the property that makes the encumbrance check robust (when it is executed by a mutation at the same time an AddEncumbrance operation is performed): a mutation that reads the encumbrance index while an AddEncumbrance call is made will be invalid and retried.

There are three valid points for certificates. First the endorsing peer checks the client's TLS certificate with the TLS-ca tls-chain on the channel. Next, the chaincode runtime verifies the signing certificate for signability against the definition of the signing certificate's MSP: root CA, intermediate CAs, OUs, CRL and only those certificates that meet the MSP's definition of the certificate identity will be sent back by ctx.GetClientIdentity(). Within the chaincode itself, a requireRole function is used to gate the chaincode while reading the role attribute from the certificate. At runtime, when the transaction is committed, the set of endorsement signatures is verified against the endorsement policy of the chaincode, in the chaincode's native language. If a signature is missing, or an unexpected signer, an expired certificate or a revoked certificate have been used — throw ENDORSEMENT_POLICY_FAILURE.

5.3 Privacy Architecture

There are two principles about privacy. Including any field which can be used to identify the user personally, this data is never stored in plain text in the ledger. But for the ledger, only the minimum is maintained to allow for integrity verification, next transition authorisation or audit support. That translates to a hashed identifier (sha256 of Aadhaar or PAN, salted with a channel-scoped salt), the parcel structural record, the IPFS content identifier of the encrypted document bundle as well as an append-only history of mutations. Figure 3 follows the same split through the write and read paths and shows where the time-limited IBE disclosure fits in. Table 5 sets out what sits on each layer and who is allowed to read it. Real property sale deeds, cadaster image, identity-proof scans and witness affidavits all reside within an institutional cluster of IPFS so those aren't uploaded.

5.3.1 Two Encryption Primitives, Two Purposes

CP-ABE is capable of dealing with long-lived documents giving its readership based on role, district and time period [8, 9, 21]. All our primitives are borrowed from the famous short-lived primitive of Shamir [22] and the Boneh Franklin construction [10] that made it practical. A policy defined as “(role=registrar AND district=Ludhiana) OR role=audit” would mean that sales in Ludhiana would be encrypted for the registrar, and for other roles as well provided they are doing audits. Only those who have an attribute key that fit the policy can decrypt, no one else. This is the proper means of access when the document must remain accessible to a body of officers, forever or for the life of the document.

The other case, short disclosure, identity and time bound, is covered by IBE. Get an Admission slip for Registrar for copying of deed to a particular Notary at notary123@reg.punjab.gov.in for 07 days. Because of access is for a short and fixed period of time and the recipient is one identity (not a role class), that is how IBE is used. ‘This person' is something that must be expressed in a rather awkward way in ABE and ‘until next Friday' is something that it simply cannot express. IBE standalone is unable to say 'any auditor in any district, indefinitely'. The two primitives do not have any overlap. They are representative of true distinct cases of real-life situations the workflow must process.

5.3.2 Key Management: Authority, Rotation, and Revocation

Importantly, key management becomes independent of the issuance of MSPs, to avoid having one globally-used trust root. The State Revenue Department has an Attribute Authority that operates an ABE key management appliance separate from the Fabric CAs, and a Private Key Generator for IBE. Whereas the role, the district, the tehsil, the valid_from and valid_until is embedded in each attribute key and issued to officials, based on the HR roster. Rotation will occur every 90 days. By using a KeyRotation chaincode event, the rotation event itself is written on chain, and thus auditors are able to correlate a Ciphertext with the current Active Attribute Generations later.

Revocation is done via attribute versioning for ABE keys. Change in an official's state causes the attribute version to be increased and the old attribute version(s) added to a small revocation list stored on-chain. The new version is for use with new encrypt texts only. When a registrar-side rewrap routine is necessary for old ciphertexts to be available for continued access, it simply re-encrypts the old ones; this mirrors proxy re-encryption schemes suggested for similar revocation-and-rotation scenarios [23]. The revocation of IBE is easier. Each token is given an explicit expiry, and this is checked when it's decrypted [10]. There's no additional maintenance — once the validity of the token ends it doesn't work anymore.

5.3.3 Search and Audit on Encrypted Records

Each record is encrypted, a two-layer index is used for searching the records. Before encryption each document is normalised for a set of standard pieces of information: Those keywords are encrypted using a deterministic searchable symmetric encryption scheme [24] (using the ABE policy keyed from the document's ABE policy) and the trapdoor is stored on-chain together

with the CID. Hence when a registrar sends a trapdoor query for "all mutations on parcel X in 2025," it provides the CIDs which match that query without ever providing the keyword to outsiders to the policy. Auditors with the audit attribute walk a different path - they decrypt the audit-keyed envelope of each document in their scope, re-create the hash chain of the documents' canon and check that the hashes provided on chain, the CIDs, and the decrypted documents' content match.

5.4 Implementation Status and Scope

To make the boundary between implemented and conceptual components plain, Table 6 lists each component along with its status. The rest of the paper can then be read without confusion about what has been built and what is described as future work. The six entries marked Implemented are sufficient to support the experiments in Section 6. The four entries marked Conceptual are the integration points where production deployment would require external action: statutory amendment, UIDAI engagement, court IT integration. They are described in this paper but not benchmarked.

6. Results and Performance Evaluation

6.1 Methodology

Benchmarks are produced with Hyperledger Caliper v0.5.0 driving the network described in Section 5.1, following the methodology used in published Hyperledger Fabric performance studies [20, 25, 26]. Each Caliper round consists of a 30-second warm-up followed by a 60-second steady-state measurement window. Send rates are swept over {50, 100, 200, 400, 800, 1200, 1600} transactions per second; each rate is repeated five times and we report mean ± standard deviation.

Table 5. On-chain / off-chain data placement

Layer	What is stored	Who can read
On-chain (ledger)	Hashed owner IDs, parcel structural record, mutation history, doc CIDs, encrypted keyword trapdoors, encumbrance flags, key-rotation events	All channel members; query results gated by chaincode role checks
Off-chain, encrypted (IPFS)	Sale deeds, cadastral maps, identity-proof scans, court orders, mutation forms	Holders of an ABE attribute key matching the policy, or a valid IBE token
Off-chain, plaintext (institutional)	HR records used to derive attributes (kept by State Revenue HR system)	HR administrators only; outside the blockchain trust boundary

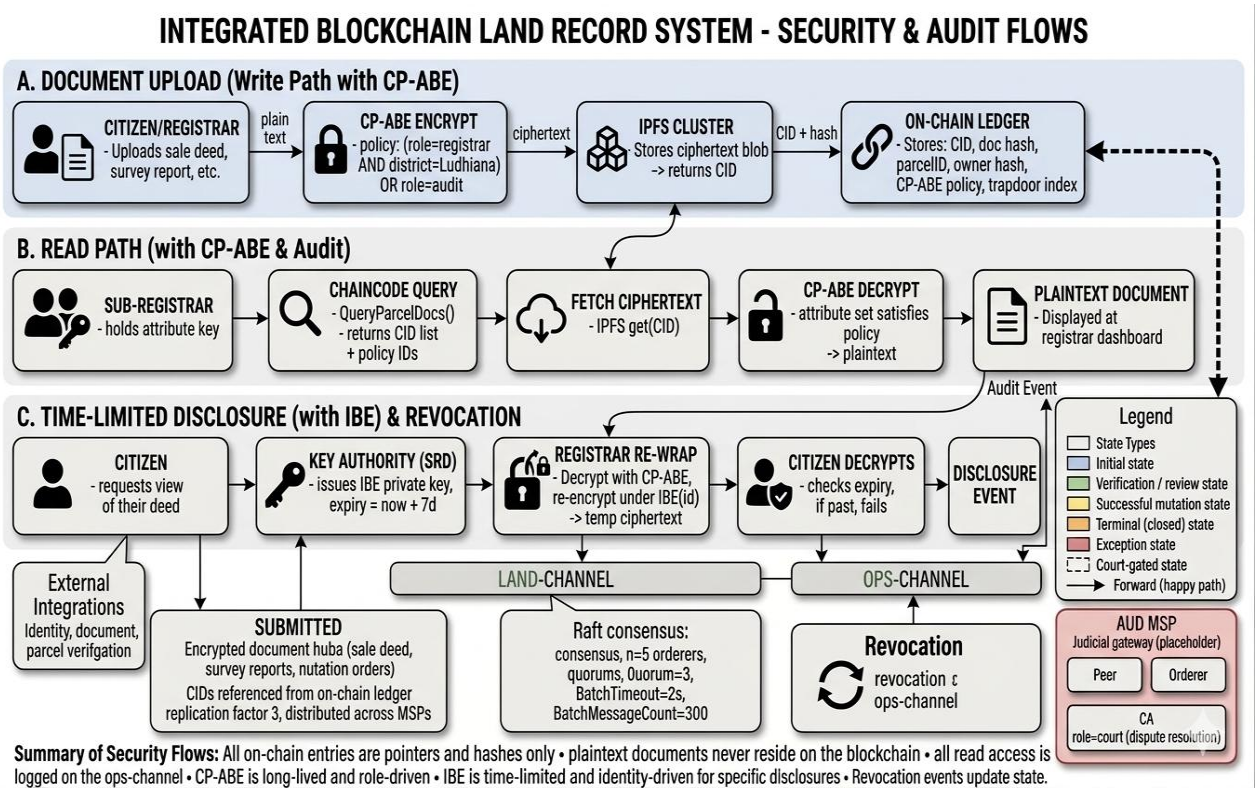

Figure 3. Privacy architecture on-chain pointers, off-chain encrypted documents

Throughput is computed as the number of transactions that reached the COMMITTED state within the measurement window divided by the window length. Latency is computed per transaction as the time at which the ordering service's leader emits the containing block (block.header.number's commit timestamp at the leader peer) minus the time at which Caliper invoked the proposal at the client. All clocks are synchronised through chrony to a common NTP source, with measured drift below 2 ms. Failed transactions, which include endorsement failures, MVCC conflicts, and timeouts, are excluded from the throughput numerator and reported separately as failure rate.

6.2 Workload

The workload division (60% query, 25% RegisterParcel, 10% TransferOwnership, 5% AddEncumbrance) was designed to reflect the daily workload distribution seen in publicly available sub-registrar volume reports for the two Punjab districts (Ludhiana and Patiala) of the period 2022 to 2024. Only published aggregate volume statistics were used, not record-level figures and the payload of the workload is synthetic but realistic of a mix. The average payload sizes, sized on proposal byte level, are: for RegisterParcel 3.21KB, for TransferOwnership 1.84KB, for AddEncumbrance 0.96KB and for QueryParcel 0.31 KB.

Table 6. Implementation status of each component.

Component	Status
Hyperledger Fabric v2.5.4 with 5 orgs, MSPs, Raft ordering	Implemented and benchmarked
land-record-cc chaincode (Go) with all functions in Table 4	Implemented
CP-ABE and IBE key services on a test-bench PKG	Implemented (Charm-Crypto v0.50)
Encrypted IPFS object store with searchable keyword index	Implemented
Web dashboards for registrar, patwari, citizen, audit	Implemented (React 18 + Fabric Gateway)
Caliper-driven benchmarking pipeline	Implemented and reproducible
Live integration with Bhulekh / Mahabhulekh / Dharani portals	Conceptual; replaced by mock connector
Aadhaar verification through UIDAI APIs	Conceptual; replaced by hashed-Aadhaar input
Court e-filing and dispute API integration	Conceptual; modelled by a Court MSP placeholder
Statutory recognition as primary record	Out of scope

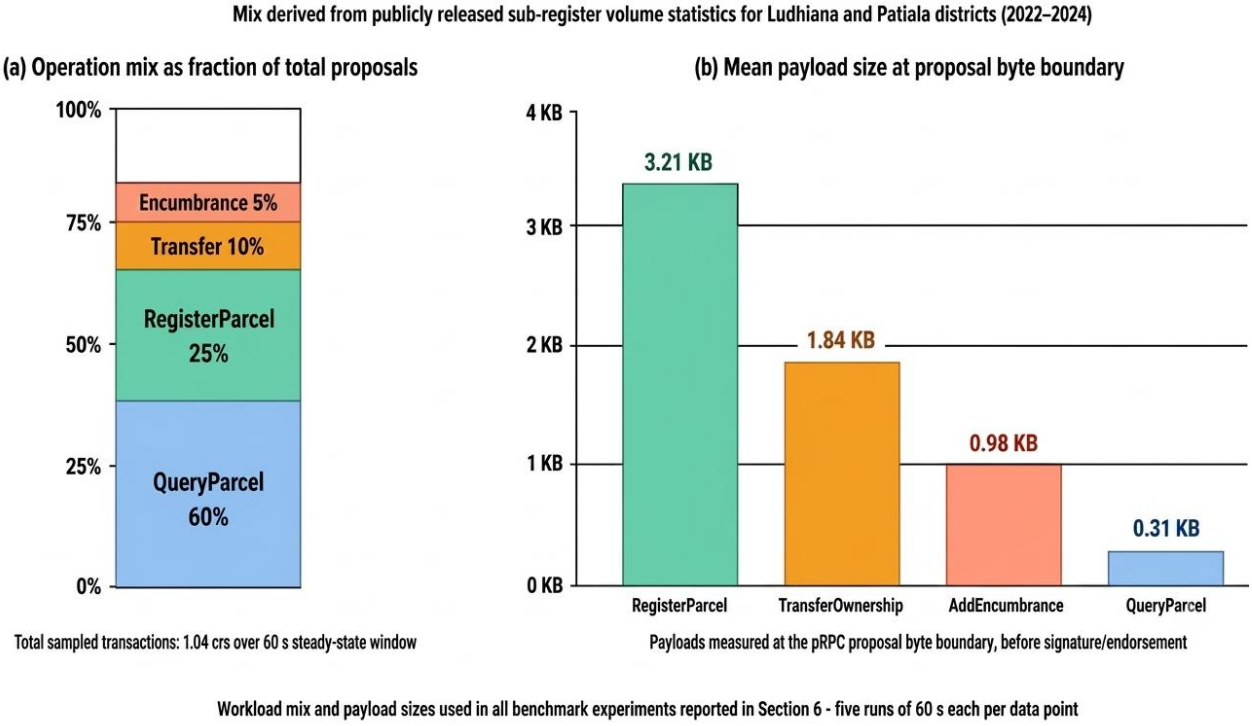

Figure 4. Workload composition and payload distribution

Table 7. Benchmark configuration as actually used in this section.

Parameter	Value
Tool	Hyperledger Caliper v0.5.0
Network	5 peers + 5 orderers + 2 client nodes, isolated 1 Gbps LAN, sub-ms RTT
Per-node hardware	8 vCPU, 16 GB RAM, NVMe SSD, Ubuntu 22.04 LTS
State DB	CouchDB 3.1, indexes on parcelID and ownerHash
Workload mix	60% Query, 25% Register, 10% Transfer, 5% Encumbrance
Mean payload	Register 3.21 KB; Transfer 1.84 KB; Encumbrance 0.96 KB; Query 0.31 KB
Endorsement count	2 (StateRevenue.peer, SubRegistrar.peer) for mutations; 1 for queries
BatchTimeout / MaxMessageCount / PreferredMaxBytes	2 s / 500 / 2 MB
Send-rate sweep	{50, 100, 200, 400, 800, 1200, 1600} tps
Round structure	30 s warm-up + 60 s steady-state; 5 repetitions per point
Throughput definition	committed-tx count / measurement window
Latency definition	leader-block-commit time − client send time, NTP-synced (< 2 ms drift)

6.3 Configuration

Table 7 collects the benchmark configuration used throughout this section, so the numbers that follow can be reproduced without working back through the earlier setup.

6.4 Throughput and Latency Results

Peak throughput on this hardware is 612 ± 18 tps at a send rate of 800 tps. Push past that and the failure rate spikes — CouchDB write contention, mostly — so committed throughput plateaus. Mean end-to-end latency at 400 tps is 1.84 ± 0.21 s, with a 95th percentile of 2.93 s. At lower send rates, latency is dominated by the ordering BatchTimeout: at 100 tps the average block is closed by timeout rather than by size (it contains roughly 200 transactions), giving an almost-constant 2-second floor. Figure 5 plots the same results, with throughput on the left and latency on the right, and the knee is easy to pick out. Table 8 has the headline numbers.

6.5 Bottleneck Analysis

Beyond a send rate of 800 tps, throughput plateaus and the failure rate rises from below 1% to 7.4%. We profiled the network at this knee. The endorsement RTT histogram is flat at 18 ms median and 41 ms p95 and does not move with send rate, which rules out endorser saturation. The fraction of blocks closed by reaching MaxMessageCount before BatchTimeout rises from 14% at 200 tps to 96% at 800 tps, indicating that the orderer is no longer the limit either. CouchDB write-queue depth, however, climbs from below 5 to over 80 across the same range, and the failures observed at 1200 tps are dominated by MVCC_READ_CONFLICT errors on the most-touched parcel keys. The bottleneck is therefore state-DB contention on hot keys, and is mitigated either by sharding the parcel key space across CouchDB instances or by switching to LevelDB at the cost of losing rich queries. The same hotspot in CouchDB-backed Fabric deployments has been reported in earlier benchmarking studies [20, 26].

We are not claiming 800 tps as a steady-state operating point. We're reporting it as the saturation knee for this hardware and configuration. Anyone deploying this in production would either provision more state-DB capacity or accept a lower steady-state target — the choice depends on how much head-room is wanted above the actual district load, which in our data is well under 50 tps even at peak.

6.6 Sensitivity to Network Conditions

Real deployments will not run on a single 1 Gbps switch. To estimate behaviour under realistic wide-area conditions, we re-ran the benchmark with one-way latency injected between orderer pairs using tc-netem. With 50 ms injected latency, peak throughput drops from 612 tps to 388 tps, and mean latency at 400 tps rises from 1.84 s to 2.41 s. With 150 ms injected latency, peak throughput falls further to 224 tps, with 4.06 s mean latency at the same send rate. Table 9 has the peak throughput and the 400 tps latency for each injected-latency setting. Even at the worst of these settings, the prototype remains comfortably above the daily transaction volumes recorded for the two Punjab districts whose workload distribution we used. The recorded peak in those districts is fewer than 50 transactions per second, even at the busiest hour of the busiest day.

Wide-area deployment is therefore feasible at the throughput levels the workload actually demands.

Table 8. Headline performance numbers (mean ± standard deviation over five runs)

Send rate (tps)	Throughput (tps)	Mean latency (s)	p95 latency (s)	Failure rate
50	49.8 ± 0.4	2.05 ± 0.08	2.31	0.0%
100	99.6 ± 0.5	2.04 ± 0.09	2.34	0.0%
200	199.1 ± 0.9	1.92 ± 0.14	2.51	0.1%
400	397.8 ± 2.2	1.84 ± 0.21	2.93	0.3%
800	611.7 ± 18.2	2.71 ± 0.46	4.62	1.4%
1200	598.4 ± 31.9	5.43 ± 1.12	9.18	7.4%
1600	562.1 ± 42.7	8.92 ± 1.87	14.32	12.1%

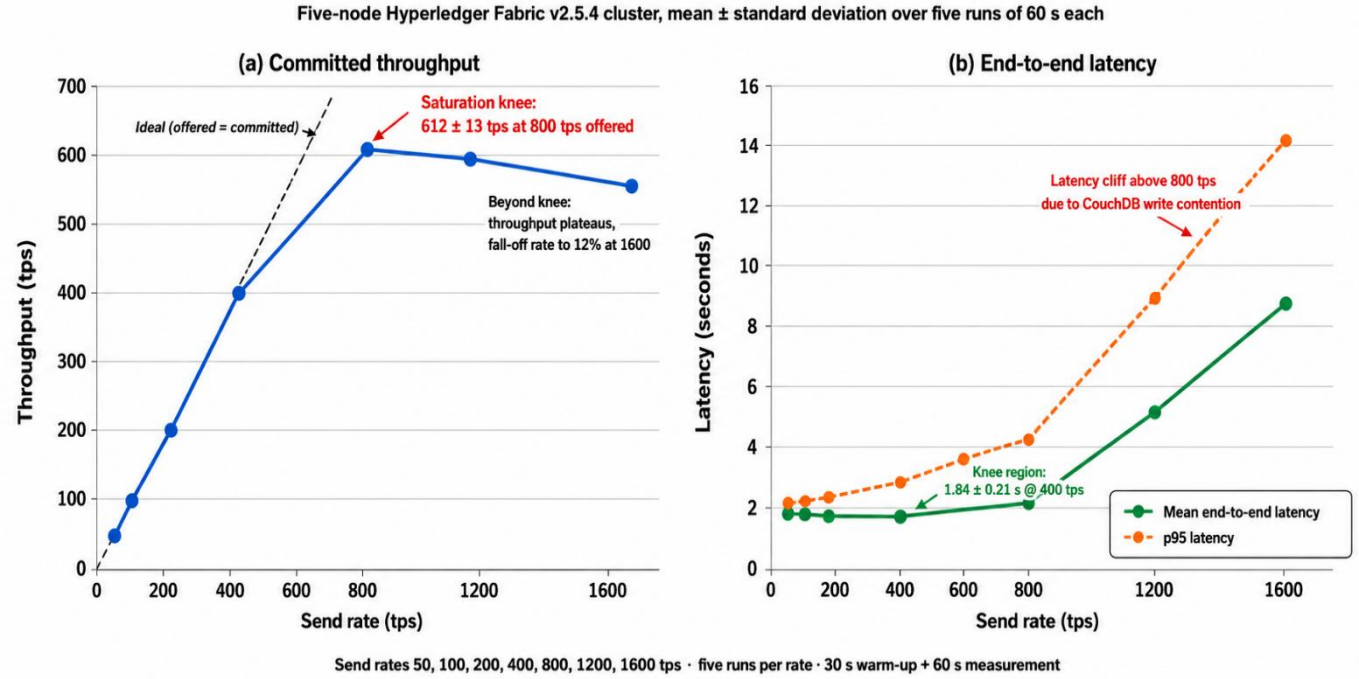

Figure 5. Throughput and latency vs send rate

Table 9. Sensitivity to inter-orderer latency

Injected one-way latency	Peak throughput (tps)	Mean latency at 400 tps (s)
0 ms (LAN)	612 ± 18	1.84
50 ms (regional WAN)	388 ± 12	2.41
150 ms (cross-country)	224 ± 9	4.06

Table 10. Reported throughput, with caveats

Study	Reported peak throughput	Caveat
Thakur *et al.* [6]	Cited from BSV literature, ~50–100 tps end-to-end	Public chain; numbers not measured in their own setup
Shinde *et al.* [12]	Not reported (survey + proposed design)	Blockchain + IPFS proposal; platform and load not stated
Krishnapriya & Sarath [13]	Approximately 350 tps on Hyperledger Fabric (single MSP)	Single MSP, simple chaincode; no document confidentiality
Sahai & Pandey [5]	Not reported (conceptual design)	Contract-level definition; no deployment or measurement
This work	612 ± 18 tps peak; 397.8 ± 2.2 sustained at 400 tps	Five-MSP permissioned; full workflow, ABE+IBE, encumbrance/dispute

6.7 Comparison with Reported Prior Numbers

Prior India-focused work has reported throughput numbers under widely differing conditions, including different chains, different hardware, and sometimes generic load.

An apples-to-apples comparison is not always possible. We therefore make a structured comparison rather than a quantitative one. Table 10 places our peak number alongside the figures reported in the studies surveyed in Section 2.2, with caveats. Two patterns emerge. The public-chain pilot [6] runs on infrastructure with roughly an order of magnitude lower throughput and substantially higher per-transaction cost than a permissioned ledger, a property of public-chain consensus and gas pricing rather than of the application. Single-MSP permissioned designs [13] achieve comparable throughput on similar hardware, but do so without the role-aware authorisation, document confidentiality, and dispute lifecycle that the present work provides. The like-for-like comparison is therefore on the operational properties rather than on the raw numbers.

7. Limitations and Practical Implications

Three things, specifically, are now demonstrated rather than asserted. First, role-aware governance through six MSPs is workable in practice the chaincode rejects what it should reject (a registrar without role=registrar in their certificate, a Court MSP signature missing on a dispute resolution) and accepts what it should accept, with no manual config changes between runs. Second, the combined CP-ABE plus IBE design is operable on official hardware. The key authority issues, rotates, and revokes attribute keys without operator intervention, and the on-chain key-rotation event log keeps the cryptographic state auditable after the fact. Third, throughput and latency, even with wide-area latency injected, sit comfortably above what a typical Indian district sub-registrar deals with day-to-day. The bottleneck under high load is the state database, not consensus or endorsement and that's the kind of bottleneck production engineering knows how to handle.

The prototype has four limitations we want to be clear about. The workload mix comes from publicly available aggregate volume data record-level traces would let us validate the mix against actual sub-registrar timestamps, but those records aren't public. The Court MSP is, at this point, a placeholder for a court-side IT integration that doesn't yet exist in the form we're modelling, in any state's e-courts deployment. The placeholder is fine for protocol-level evaluation, but real integration would need court-system reform. Aadhaar handling here stops at hashed identifiers; integrating UIDAI's authentication APIs would change the trust model, and we've left that as future work. Finally, our wide-area sensitivity numbers come from synthetic latency injection, not from deploying across a real geographically distributed network. The synthetic numbers are probably a lower bound on how hard real deployment will be.

Internal validity is constrained mainly by the closed test environment: we control the network and the workload, which removes confounds but limits external validity. External validity is addressed in part by the wide-area sensitivity study and in part by the use of an India-derived workload mix; we encourage replication on different state schemas. Construct validity rests on the alignment of our chaincode states with the actual administrative workflow. The state machine was developed in consultation with a small sample of practitioners; that consultation was small in scale, and a more systematic field validation is desirable before any production deployment is considered.

For a state revenue department thinking about a permissioned blockchain layer, three things follow from the work above. The role-aware MSP design is what gives you an auditable chaincode in the first place. Collapse the workflow into a single MSP and you lose that property — the system devolves into a tamper-evident database. Document confidentiality isn't optional in the Indian context, either: the on-chain layer needs to hold pointers and hashes only, with document content encrypted off-chain [27], and the encryption layer has to handle both role-driven and identity-and-time-bound access. One mode isn't enough. The integration points with state portals (Bhulekh, Mahabhulekh, Dharani) and with UIDAI are where the real engineering effort lies. The chain layer itself, evaluated in this paper, is the smaller part of the problem.

Two corollaries fall out of the bottleneck analysis in §6.5. The multi-MSP endorsement policy adds essentially no measurable cost over a single-MSP setup the limit on this hardware is CouchDB write contention on hot parcel keys, not signature collection [20]. And throughput stays comfortably above district-level demand even at 150 ms one-way latency between orderers, which means a wide-area deployment across state offices is feasible without re-engineering the consensus path. Where production deployment will actually need engineering effort is the state-DB layer (sharding parcel keys, or moving to LevelDB and accepting the loss of rich queries) and the integration glue with state portals [4, 28]. Not the blockchain core.

8. Conclusion and Future Work

This paper described and gave an evaluation of the Permissioned blockchain prototype that is designed for Indian land record as a verification and audit layer which can be added over the statutory record and not as a replacement. The prototype realizes six Administrative roles, defined as separate Membership Service

Providers, manages mutations and disputes via role aware endorsement policies and provides documents' confidentiality protection through a hybrid CP-ABE and IBE privacy model. The publicly available volume information from sub-registrars to calculate the workload volume reveals that the system would support any typical district in India with a decent margin of safety, even under realistic wide-area latency considerations. This study noted three directions for future work. First, it should be interfaced with the state land-record websites like the Bhulekh website, Mahabhulekh, and Dharani, by using schema mapping, and by developing event-driven bridges. Technically possible but an engineering challenge. Second, the prototype should be tested along with an actual sub-registrar office with various operational parameters such as an operator's workload, time taken to complete a task, error rate, and efficiency of the process as applied in the proposed workflow. Third, more legal and policy work is needed to define the evidentiary effects that the entries in the blockchain would have. This question is not technical and not part of the scope of a technical study, but is nevertheless crucial for production level deployments. Thus, it seemed that the present prototype should be rather thought of as a proof-of-concept of how role-aware permissioned ledgers with suitable privacy-preserving features can fit into the existing land-record workflow in India. The results indicate that although the key engineering issues are limited and feasible to solve, key issues remain for statutory recognition, administrative uptake and inter-departmental integration.

References

[1] Government of India. (1908). The Registration Act, 1908 (Act No. 16 of 1908). India Code. Retrieved from https://www.indiacode.nic.in/handle/123456789/2406

[2] Government of India, Ministry of Rural Development. (2008). Digital India Land Records Modernization Programme (DILRMP) guidelines. Department of Land Resources.

[3] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S.W. Cocco, J. Yellick, Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In Proceedings of the Thirteenth EuroSys Conference, (2018) 1-15. https://doi.org/10.1145/3190508.3190538

[4] S. Kumar, M. Kumar, C.N. Azmea, K.K. Vaigandla, BCSDNCC: a secure blockchain SDN framework for IoT and Cloud Computing. International Research Journal of Multidisciplinary Technovation, 6(3), (2024) 26-44. https://doi.org/10.54392/irjmt2433

[5] A. Sahai, R. Pandey, Smart Contract Definition for Land Registry in Blockchain. In 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), IEEE, Gwalior, India, (2020) 230–235. https://doi.org/10.1109/CSNT48778.2020.9115752

[6] V. Thakur, M.N. Doja, Y. K. Dwivedi, T., Ahmad, G. Khadanga, Land Records on Blockchain for Implementation of Land Titling in India. International Journal of Information Management, 52, (2020) 101940. https://doi.org/10.1016/j.ijinfomgt.2019.04.013

[7] H. Mukne, P. Pai, S. Raut, D. Ambawade, Land record management using Hyperledger Fabric and IPFS. In 2019 10th International Conference on Computing. Communication and Networking Technologies (ICCCNT), IEEE, (2019) 1-8 https://doi.org/10.1109/ICCCNT45670.2019.8944471

[8] J. Zhang, A. Datta, Blockchain-Enabled Data Governance for Privacy-Preserved Sharing of Confidential Data. PeerJ Computer Science, 10, (2024) e2581. https://doi.org/10.7717/peerj-cs.2581

[9] J. Bethencourt, A. Sahai, B. Waters, (2007) Ciphertext-Policy Attribute-Based Encryption. In 2007 IEEE Symposium on Security and Privacy (SP'07), IEEE Computer Society. https://doi.org/10.1109/SP.2007.11

[10] D. Boneh, M. Franklin, Identity-Based Encryption from the Weil Pairing. SIAM Journal on Computing, 32(3), (2003) 586–615. https://doi.org/10.1137/S0097539701398521

[11] S. Nakamoto, (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Available at: https://bitcoin.org/bitcoin.pdf

[12] D. Shinde, S. Padekar, S. Raut, A. Wasay, S.S. Sambhare, Land Registry Using Blockchain: A Survey of Existing Systems and Proposing a Feasible Solution. In 2019 5th International Conference on Computing, Communication, Control and Automation (ICCUBEA), IEEE, Pune, India, (2019) 1–6. https://doi.org/10.1109/ICCUBEA47591.2019.9129289

[13] S. Krishnapriya, G. Sarath, Securing Land Registration using Blockchain. Procedia Computer Science, 171, (2020) 1708–1715. https://doi.org/10.1016/j.procs.2020.04.183

[14] NITI Aayog. (2020). Blockchain: The India strategy – Towards enabling ease of business,

ease of living, and ease of governance, Part I. Government of India. Retrieved from https://niti.gov.in/sites/default/files/2020-01/Blockchain_The_India_Strategy_Part_I.pdf

[15] D. Ongaro, J. Ousterhout, In Search of an Understandable Consensus Algorithm. In Proceedings of the 2014 USENIX Annual Technical Conference (USENIX ATC' 14), (2014) 305–319.

[16] L. Lamport, The Part-Time Parliament. ACM Transactions on Computer Systems, 16(2), (1998) 133–169. https://doi.org/10.1145/279227.279229

[17] M. Castro, B. Liskov, Practical Byzantine Fault Tolerance. In Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI' 99), (1999) 173-186.

[18] S.S.P. Pennada, S.K. Nayak, V.K.M, Insider Threat Detection using Behavioural Analysis through Machine Learning and Deep Learning Techniques. International Research Journal of Multidisciplinary Technovation, 7(2), (2025) 74–86. https://doi.org/10.54392/irjmt2527

[19] D. Rekha, K. Baalaji, Deep Learning-based Secure Routing Framework for Blockchain-Enabled Autonomous Military Wireless Sensor Networks. International Research Journal of Multidisciplinary Technovation, 8(2), (2026) 92–108. https://doi.org/10.54392/irjmt2625

[20] P. Thakkar, S. Nathan, B. Viswanathan, (2018) Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform. In 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), IEEE, Milwaukee, WI, USA. https://doi.org/10.1109/MASCOTS.2018.00034

[21] A. Sahai, B. Waters, Fuzzy Identity-Based Encryption. In: Cramer, R. (Eds) Advances in Cryptology – Eurocrypt 2005. Eurocrypt 2005. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 3494, (2005) 457-473. https://doi.org/10.1007/11426639_27

[22] A. Shamir, (1985). Identity-Based Cryptosystems and Signature Schemes. In: Blakley, G.R., Chaum, D. (eds) Advances in Cryptology. CRYPTO 1984. Lecture Notes in Computer Science, 196. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-39568-7_5

[23] R. Patil, Y.H. Patil, D. Naik, R. Gangarde, A. Joshi, A. Bannore, Secure Firmware Over the Air Updates for Vehicles using Blockchain, Signcryption, and Proxy Re-Encryption. International Research Journal of Multidisciplinary Technovation, 7(3), (2025) 383–396. https://doi.org/10.54392/irjmt25327

[24] D.X. Song, D. Wagner, A. Perrig, Practical Techniques for Searches on Encrypted Data. In Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P' 00), IEEE, (2000) 44–55. https://doi.org/10.1109/SECPRI.2000.848445

[25] Hyperledger Foundation. (2023). Benchmarking Hyperledger Fabric 2.5 Performance. Linux Foundation Decentralized Trust. https://www.lfdecentralizedtrust.org/blog/2023/02/16/benchmarking-hyperledger-fabric-2-5-performance

[26] M. Abbasi, J. Silva, P. Vaz, A. Soares, P. Martins, (2025). Performance benchmarking of Hyperledger Fabric networks: Insights for scalability and optimization. In J. L. Reis, M. K. Peter, L. P. Reis, & Z. Bogdanovic (Eds.), Marketing and Smart Technologies. ICMarkTech 2023. Smart Innovation, Systems and Technologies, 393, 1-14. Springer. https://doi.org/10.1007/978-981-97-3698-0_1

[27] G. Zyskind, O. Nathan, (2015) A. Pentland, Decentralizing Privacy: Using Blockchain to Protect Personal Data. In 2015 IEEE Security and Privacy Workshops, IEEE, USA. https://doi.org/10.1109/SPW.2015.27

[28] L. Liu, K. Omote, Efficient Authentication System Based on Blockchain for E-government. PLOS ONE, 20(12), (2025) e0336997. https://doi.org/10.1371/journal.pone.0336997

Authors Contribution Statement

Aseem Khanna: Conceptualization, Methodology, Software, Investigation, Writing – Original Draft. Pritpal Singh: Methodology, Validation, Writing – Review & Editing. Prikshat Kumar Angra: Methodology, Formal Analysis, Visualization, Writing – Review & Editing. All authors have read and agreed to the published version of the manuscript.

Funding

The authors declare that no funds, grants or any other support were received during the preparation of this manuscript.

Competing Interests

The authors declare that there are no conflicts of interest regarding the publication of this manuscript.

Data Availability

The chaincode, configuration files, benchmark settings, and aggregated benchmark results supporting the findings of this study are available from the

corresponding author on reasonable request. The workload distribution used for benchmarking is derived from publicly available aggregate sub-registrar volume reports and does not contain personally identifiable information.

Has this article screened for similarity?

Yes

About the License