IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511193

SecureChain: Advancing Document Integrity Through Decentralized Intelligence

Govindraj B. Chittapur[1], Shreeraksha P. Kulkarni[2], Spoorti Vitthal Yalagur[3], Spoorti Vijaykumar Ganiger[4], Subhashini Balappa Teli[5]

Dept. of Computer Science and Engineering, Basaveshwar Engineering College Bagalkote, Karnataka, India[1-5]

Abstract: Verifying the authenticity of digital academic documents is slow and error-prone when it depends on manual contact with the issuing institution or on a single central database, and a digital file can be altered without visible trace. This paper presents SecureChain, a role-based web platform that combines blockchain registration, decentralized storage and AI-based document analysis to verify academic documents. When an institution registers a document, the backend computes its SHA-256 digest, encrypts the file with AES-256, stores it on IPFS through Pinata, and records the document identifier, digest and content identifier (CID) in a Solidity smart contract. A company submits a verification request for a student-provided document, and an authorized faculty verifier checks it in four stages (AI-based forgery analysis, SHA-256 hash comparison, blockchain record verification and college record verification) before issuing an RSA-SHA256 signed report. The prototype uses Node.js, Express, MongoDB, Hardhat, Viem and the Ethereum Sepolia test network. In testing, all 20 functional and 8 security test cases passed, and a modified marks card was rejected through a hash mismatch. Average component timings were 15 ms for hashing, 3.5 s for IPFS upload and 4.0 s for blockchain registration. In a preliminary evaluation on 42 documents (32 genuine, 10 modified), the ML-based forgery detector reached 90.48% accuracy, 75.00% precision, 90.00% recall and 81.82% F1-score.

Keywords: blockchain, document verification, IPFS, smart contract, SHA-256, role-based access control, forgery detection

I. INTRODUCTION

Academic documents such as marks cards, degree certificates and transcripts are now issued, stored and exchanged mostly as digital files. Employers and other organizations that receive them still need to know whether a file is genuine and unaltered. In practice this is often done by contacting the issuing institution, which is slow, depends on staff availability and scales poorly. A digital file can also be edited, for example by changing marks or a registration number, in a way that is hard to notice by inspection. The cost of such forgery falls on employers who must repeat checks, on institutions that answer verification calls, and on students whose genuine records are doubted.

A widely studied remedy keeps the document off the blockchain, usually on IPFS, and records only a cryptographic hash and a storage reference on-chain, where later changes are easy to detect [1], [2], [3]. Comparing the hash of a submitted file with the registered hash then shows whether the file has changed. Hash anchoring alone, however, does not protect confidentiality, does not control who may request verification, does not examine the visual content of a document, and does not produce an accountable decision that can be shown to a third party.

A. Problem statement and objectives

The problem addressed is how to let an organization check a student-provided academic document against the institution's trusted record without contacting the institution each time, while keeping the document confidential, restricting who can request verification, and leaving an accountable result. The objectives are to:

1) register genuine academic documents with an integrity-protecting record that cannot be silently changed;
2) keep documents off-chain in encrypted form and store only compact references on-chain;
3) restrict operations by role, so that students, companies and verifiers see only what they are permitted to see;
4) detect any modification of a registered document by comparing hashes; and
5) give a human verifier supporting evidence and issue a digitally signed verification report.

B. Contributions

This paper presents SecureChain, a web-based prototype that combines these concerns in one workflow. Its contributions are:

1) a role-based architecture for four stakeholders (Institution, Student, Company and Verifier), with explicit document provision to companies and administrator approval of faculty verifiers;

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511193

2) a registration pipeline that computes a server-side SHA-256 digest, encrypts institutional documents with AES-256, stores them on IPFS and records identifier, digest and CID in a Solidity smart contract, developed on a local Hardhat network and integrated with the Ethereum Sepolia test network;
3) a four-stage verification procedure that ends in a human verifier's decision and an RSA-SHA256 signed report; and
4) An evaluation of the prototype through functional, security and performance testing, along with an AI-based document forgery-analysis module that uses Machine Learning techniques to examine document characteristics and assist the verification process, together with a preliminary evaluation of that module on labelled documents.

Section II reviews related work, Section III describes the design, Section IV the implementation, Section V the results and discussion, and Section VI concludes.

II. RELATED WORK

A. Blockchain-based document verification

Public-blockchain systems anchor document hashes on Ethereum-family networks. Nizamuddin et al. [1] combine Ethereum and IPFS for document version control with approver roles. Chowdhary et al. [4] link each certificate to a student identifier and a secret phrase in addition to the hash. Verifi-Chain [5] stores credentials on IPFS, writes the CID on-chain and requires applicant approval before a company can inspect them. EduDocs [6] hashes documents on the client and uses Pinata-hosted IPFS, and e-Registry [7] reports Sepolia benchmarks with a 1.3 s verification latency.

Permissioned and authority-based designs trade public transparency for control. BlockIPFS [8] builds a forensic access trail on Hyperledger Fabric, TVS [9] verifies office documents on Fabric, and DocCert [10] uses Merkle trees for foreign credentials. Proof-of-stake and proof-of-authority variants appear in [2], [3] and, with Merkle-root batching, in Go-Chain [11]. Layer-2 and privacy-oriented systems include Polygon-based document and diploma verification [12], [13], a self-sovereign-identity framework [14] and a zero-knowledge system, ZKBAR-V [15], that reports 5.8 ms verification latency and roughly 94% lower gas cost than its Ethereum Mainnet baseline.

B. AI-based forgery detection and storage privacy

Maamouli et al. [16] train a CNN on 232 real and 373 forged administrative PDFs and report 73.95% precision and 97.3% recall. This line of work is independent of blockchain and does not provide a trusted record of the original document. A separate study of IPFS [17] shows that request patterns on the network can reveal which data a node is interested in, so off-chain storage is not automatically private.

TABLE I SUMMARY OF SURVEYED SYSTEMS

Ref.	Platform / consensus	Storage	Privacy / security feature	Domain
[1]	Ethereum (Solidity)	IPFS; hash on-chain	Developer and approver roles	Document version control
[2]	Private blockchain (PoS)	Private IPFS; hash on-chain	Token-based sharing, access logging	Academic
[3]	Private blockchain (PoA)	IPFS and blockchain	Institution-controlled access	Academic credentials
[4]	Ethereum	IPFS and blockchain	Unique ID and secret phrase	Academic certificates
[5]	Ethereum	IPFS and Ethereum	Role-based access, applicant approval	Academic credentials
[6]	Ethereum	Pinata IPFS, MySQL, hash	Hash-only sharing with third parties	Academic
[7]	Ethereum Sepolia	On-chain hash and metadata	Client-side SHA-256	General documents
[8]	Hyperledger Fabric (permissioned)	IPFS; metadata on-chain	Forensic access audit trail	Forensics, general data
[9]	Hyperledger Fabric	Fabric ledger	Category-based grouping	Office documents
[10]	Permissioned blockchain	Merkle trees and signatures	Non-repudiation	Foreign credentials
[11]	Custom public chain (PoA)	Merkle-root batching	P-256 signatures	Government e-documents



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511193

Ref.	Platform / consensus	Storage	Privacy / security feature	Domain
[12]	Polygon (Layer 2)	IPFS and Polygon	Metadata-based verification	Administrative documents
[13]	Polygon	IPFS and blockchain	Verifier-side hash not stored	Diplomas
[14]	DLT with DIDs and verifiable credentials	Credential data held by holder	Selective disclosure	Financial and general
[15]	Dual zkEVM (Polygon)	Private zkEVM, public proofs	Zero-knowledge proofs, DIDs	Academic records
[16]	No blockchain (CNN model)	Local dataset	—	Administrative documents

C. Observations and gap addressed

Table I shows five recurring patterns. First, the hybrid design of off-chain storage with an on-chain hash is the norm. Second, the choice of platform trades transparency against control, from public Ethereum to permissioned Fabric. Third, privacy is handled unevenly: some systems add selective disclosure or zero-knowledge proofs, while others rely on hash-only sharing, and request metadata on IPFS can leak [17]. Fourth, most evaluations use testnets, small prototypes or few users. Fifth, AI-based forgery analysis is rarely combined with a blockchain record.

In the studies we reviewed, we did not find one that combines role-based provision of documents to companies, encrypted off-chain storage, an AI analysis stage and a digitally signed verifier report in a single workflow. SecureChain is an integrated prototype of that combination. We do not claim that any single component is new.

III. SYSTEM DESIGN

A. Design goals

The design follows five goals: integrity (ensures that modifications to a registered document can be detected by comparing its SHA-256 hash with the corresponding hash stored on the blockchain), confidentiality (ensures that sensitive academic documents are protected through encryption during storage and that access is restricted to authorized users), access control (ensures that each stakeholder, including the Institution, Student, Company and Verifier, can perform only the operations permitted by their assigned roles), accountability (ensures that verification decisions are associated with the authorized faculty verifier and that verification reports provide a record of the verification outcome) and honest failure handling (ensures that errors occurring during document processing, IPFS operations or blockchain transactions are properly reported rather than being treated as successful operations).

B. Roles and workflow

SecureChain has four roles (Table II). The Institution registers genuine documents, a Student manages personal documents and provides them to companies, a Company submits verification requests, and a Verifier, a faculty member approved by the Institution, reviews each request. A typical flow is: the Institution registers a marks card; the Student provides it to a Company; the Company submits a request; the Verifier reviews the evidence and records a decision; and the Company reads the signed result.

TABLE II ROLES AND RESPONSIBILITIES

Role	Main responsibilities
Institution (College Admin)	Registers genuine academic documents; views registered documents; approves or rejects faculty verifier accounts; opens an audit dashboard.
Student	Uploads, views and replaces documents; provides documents to companies; tracks verification status.
Company	Finds a student; selects a provided document; submits a verification request; views completed results.
Verifier	Reviews requests and verification evidence; records the final decision; the report is signed with RSA-SHA256.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511193

C. Architecture

Fig. 1 shows the layered architecture. A web interface serves the four roles. An authentication and authorization layer (bcrypt password hashing, JWT sessions and role-based access control) protects the Node.js/Express backend, which coordinates MongoDB (users, document metadata, sharing and verification records), the document-processing service (SHA-256), IPFS storage through Pinata, the DocumentRegistry smart contract, the AI analysis module and the verification module.

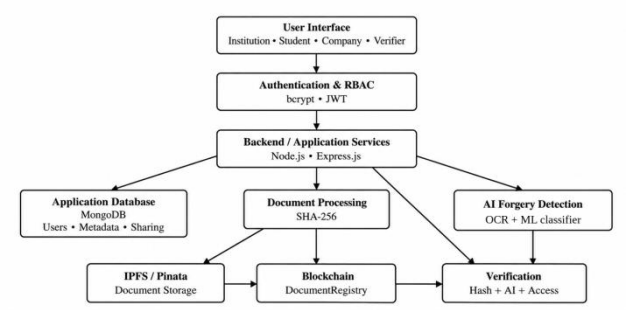

Fig. 1 SecureChain system architecture.

D. Data design

Application data is kept in MongoDB (Fig. 2). A user has one role; a document belongs to a student and records its type, SHA-256 hash, IPFS CID, metadata and uploader; a sharing record links a document, a student and a company with a permission, status and expiry date; and a verification record links a document to a verifier and stores the current hash, the result, the AI result, the report path and the digital signature.

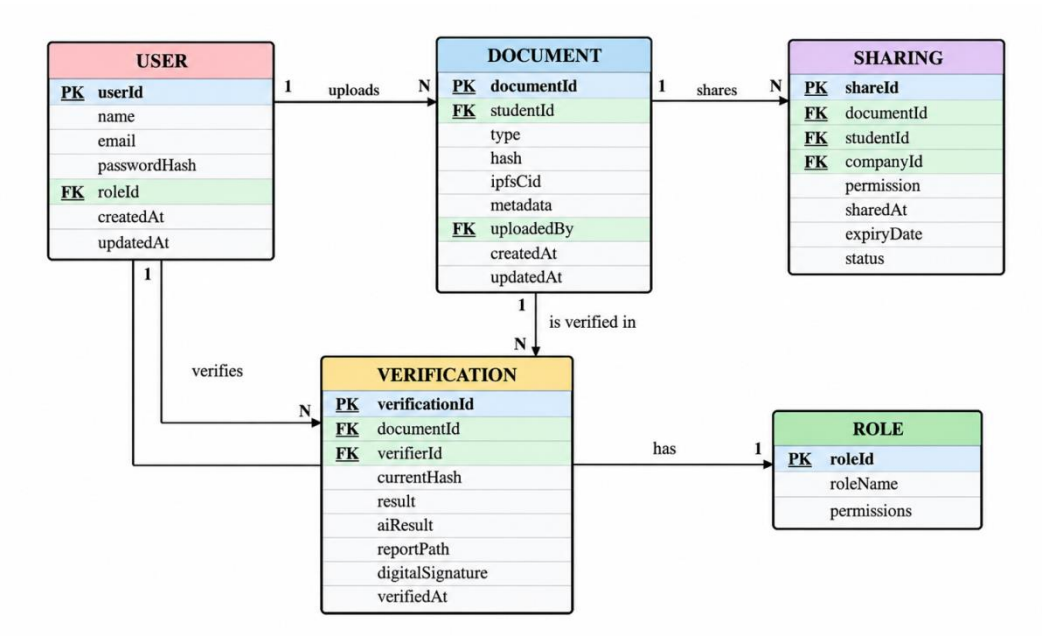

Fig. 2 Entity-relationship design of the application database.

E. Document registration

Registration is performed by the Institution (Algorithm 1). The digest H = SHA-256(D) of the original file D is computed by the backend, not on the client. According to the design in the project report, the digest is computed before encryption and the encrypted file is what is stored on IPFS.

F. Algorithm 1: Document registration

1. Authenticate the Institution user and verify the assigned role.
2. Receive the academic document and validate its file type and size (maximum 5 MB).
3. Compute the SHA-256 hash of the original document using the Node.js crypto module.
4. Encrypt the document using AES-256 encryption with the configured initialization vector (IV).



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511193

5. Upload the encrypted document to IPFS through Pinata and obtain its Content Identifier (CID).
6. Register the document identifier, SHA-256 hash and IPFS CID in the DocumentRegistry smart contract.
7. Store the corresponding document metadata in MongoDB and return the registration status.

G. Smart contract

The DocumentRegistry contract, written in Solidity, stores for each document its identifier, SHA-256 digest and IPFS CID, and allows the record to be retrieved by identifier. Complete documents are never stored on-chain, which keeps transactions small and avoids publishing document contents. The contract rejects invalid and duplicate registrations (test TC-15 in Section V), so a registered record cannot be silently replaced by a second registration.

H. Verification

A company request is reviewed by a verifier through four stages (Algorithm 2, Fig. 3). Let H be the registered digest and $H' = \text{SHA-256}(D')$ the digest of the document under verification D', which the backend retrieves from IPFS using its CID. The hash stage passes only if $H' = H$, so any change to the file contents produces a mismatch. The blockchain stage checks that the document ID and hash match the on-chain record, and the college-record stage checks that a trusted college record is linked to the request. The AI stage produces additional evidence that the verifier reviews; it does not decide the outcome by itself.

I. Algorithm 2: Document verification

1. Authenticate the Verifier and validate the assigned role and permissions.
2. Retrieve the verification request and associated document metadata, including the document identifier, IPFS CID and registered SHA-256 hash, from MongoDB.
3. Obtain the document submitted for verification and perform the required document preprocessing.
4. Compute the SHA-256 hash of the document being verified and retrieve the corresponding registered hash from the DocumentRegistry smart contract.
5. Compare the computed hash with the registered hash to determine whether the submitted document matches the institution-registered version.
6. Validate the blockchain record using the document identifier and registered hash, and check the corresponding institutional record.
7. Perform AI-based forgery analysis using Machine Learning techniques and present the analysis results to the authorized Verifier.
8. Evaluate the verification evidence and record the Verifier's final decision (Approved or Rejected).
9. Generate the verification report and apply the digital signature, if supported by the implemented signing mechanism.
10. Store the verification result, timestamp, Verifier details and generated report reference in MongoDB.

J. Signed verification report

Each completed verification generates a structured report containing the document details, student and company information, individual verification results, and the final decision recorded by the authorized faculty verifier. The report presents the outcomes of AI-based forgery analysis, SHA-256 hash comparison, blockchain record verification and institutional record validation. It also includes the verifier's name, designation, verification timestamp and digital-signature details. The report is digitally signed using the RSA-SHA256 mechanism to support authenticity and integrity verification. Any subsequent modification to the signed report can be detected through digital-signature validation.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511193

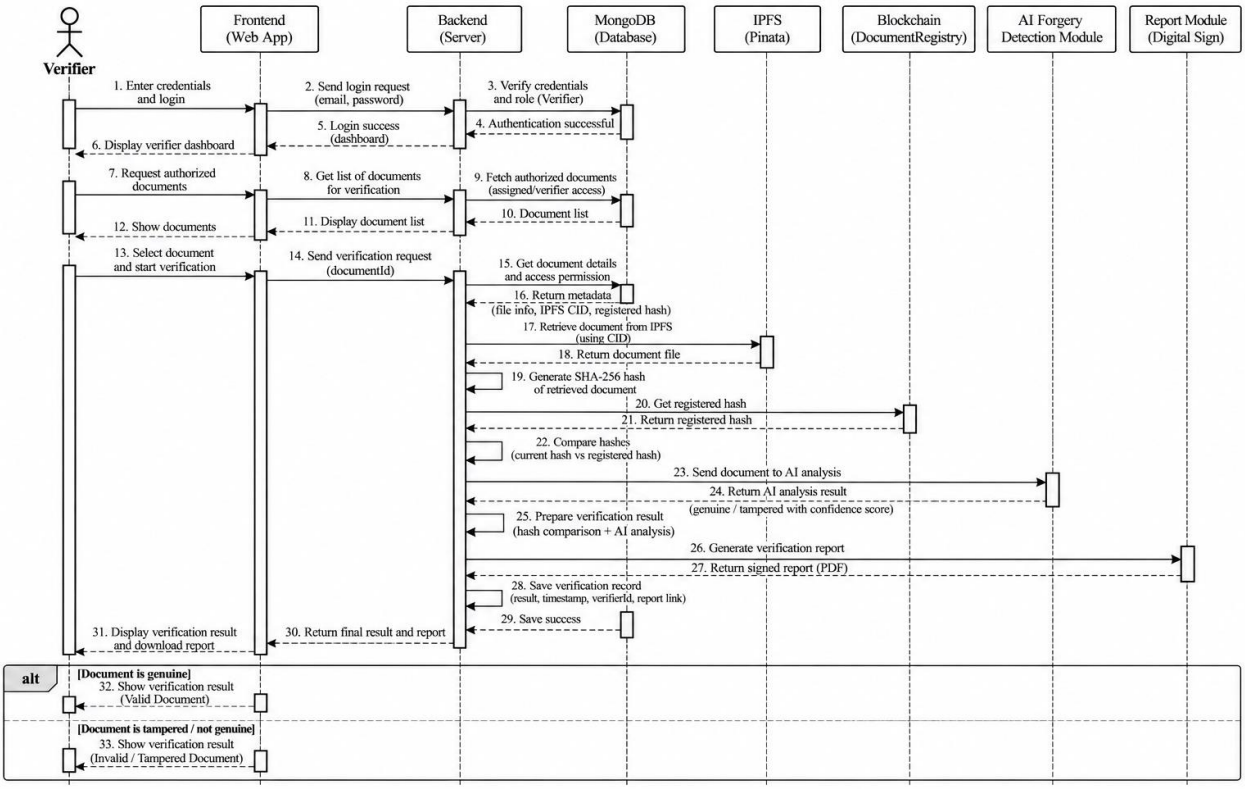

Fig. 3 Sequence of a verification request handled by a verifier.

K. Security design and threat model

Fig. 4 summarizes the five security layers. Passwords are stored as bcrypt hashes and sessions use JWT. For a user u requesting operation p, access is granted only if p belongs to the permission set of the user's role, i.e., Access(u, p) = 1 if p ∈ Permissions(Role(u)) and 0 otherwise. Institutional academic documents are encrypted with AES-256 in CBC mode with a random 16-byte IV, and the key is supplied through environment configuration. Encryption provides confidentiality, while the hash and the blockchain record provide integrity. Table III maps the main threats to mitigations and to the test cases that exercised them.

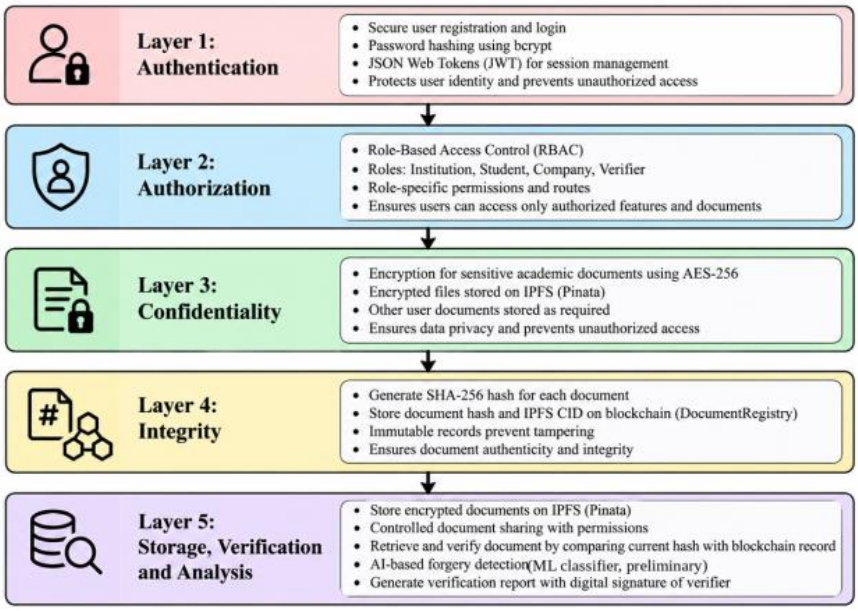

Fig. 4 Layered security design of SecureChain.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511193

TABLE III THREATS, MITIGATIONS AND TEST EVIDENCE

Threat	Mitigation	Tests
Modified document submitted	SHA-256 comparison with the registered hash	TC-18, TC-19, SR-03
Unauthenticated or cross-role access	JWT authentication; role-based access control	TC-04, TC-05, SR-01
Company reads a document without permission	Explicit provision and permission check	TC-20, SR-02
Reading sensitive academic content	AES-256 encryption; authorization	SR-07
Credential exposure in the database	bcrypt password hashing	None recorded
Silent IPFS or chain failure	Failures reported, not shown as success	SR-05, SR-06
Tampered verification report	RSA-SHA256 signature validation	SR-08
Malformed or invalid input	Input validation and rejection	TC-08, SR-04
Duplicate or invalid registration	Contract validation rules	TC-15

L. AI-assisted forgery analysis (preliminary)

A hash comparison detects any change to a registered file, so the AI stage is intended for cases where a hash alone is less informative, such as a re-saved, re-scanned or reformatted copy, and to give the verifier visual evidence. This role is a design intention; a preliminary evaluation is reported in Section V-C.

The current dataset consists of 32 genuine semester marks cards collected from approximately eight students (four semesters each). The PDFs were converted to images, and Tesseract OCR 5.5.3 extracted text from all 32 documents. Image brightness and sharpness were also inspected, and 10 modified documents were used as the forged class in the preliminary evaluation of Section V-C. Table IV summarizes the dataset and method. Because of the small sample size, a student-level split is planned so that a genuine card and its modified copy are never separated across training and test sets, and grouped cross-validation may be preferable to a single split.

TABLE IV AI DATASET AND METHOD

Item	Status
Genuine samples	32 marks cards, about 8 students, semesters 1–4 (collected)
Modified samples	10 modified documents used as the forged class in the reported evaluation
Total	42 documents (32 genuine, 10 modified) in the reported evaluation
Features	Brightness, contrast, sharpness, edge density, texture, OCR-derived features
Classifiers	Random Forest (primary); SVM (possible comparison)
Metrics	Accuracy, precision, recall, F1-score, confusion matrix, from actual predictions

A preliminary evaluation on 42 documents is reported in Section V-C. In the prototype the AI stage appears in each report as completed, and its output is reviewed by the authorized verifier.

IV. IMPLEMENTATION AND SETUP

A. Environment and technology stack

Table V lists the technology stack. Smart-contract development and unit testing used the Hardhat local network, and blockchain integration used the Ethereum Sepolia test network through an Alchemy RPC endpoint, with MetaMask for wallet operations and Viem for backend contract calls. The two environments are kept distinct in the results.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511193

TABLE V TECHNOLOGY STACK

Component	Technology
Backend	Node.js 24.15.0, Express.js
Database	MongoDB (local instance)
Authentication	bcrypt, JWT, role-based access control
Hashing / signing	SHA-256 (Node.js crypto); RSA-SHA256 report signature
Encryption	AES-256-CBC, random 16-byte IV
Smart contract	Solidity, Hardhat 3.12.0
Blockchain access	Viem, Alchemy RPC, MetaMask
Networks	Hardhat local (development, contract tests); Ethereum Sepolia (integration)
Storage	IPFS via Pinata; maximum upload size 5 MB
AI tooling	Python 3.13.7, Tesseract OCR 5.5.3, OpenCV
Platform	Windows; Intel Core i5 (12th gen), 16 GB RAM, 512 GB SSD

The hardware row gives the expected system configuration.

B. Backend, storage and blockchain integration

The backend receives an uploaded file, reads it in the document service, computes the SHA-256 digest with the Node.js crypto module, uploads the file to IPFS through Pinata and then registers the hash and CID on the blockchain through Viem. On the local Hardhat network the DocumentRegistry contract is compiled, deployed and unit-tested, with eight tests passing in one execution of the suite. On Sepolia the same contract is reached through Alchemy, and wallet operations use MetaMask. Because a test network behaves differently from a public production network in cost and confirmation time, the Sepolia results should not be read as Mainnet performance.

C. User interface

The login page lets a user choose a role. Each role then has its own dashboard: the Institution registers documents, views registered documents, manages verifiers and opens an audit dashboard; the Student manages documents and checks verification status; the Company finds students and tracks requests and results; and the Verifier reviews incoming requests and sees a summary of verified and rejected documents.

D. Running prototype

Fig. 5 shows the prototype running in the local development environment. The Hardhat node serves a local JSON-RPC endpoint and logs the deployment of the DocumentRegistry contract, which used 723,284 gas in block 1 (Fig. 5a). The contract is deployed to the localhost network through Hardhat Ignition as DocumentRegistryModule (Fig. 5b). The backend starts on port 5000 and reports a successful MongoDB connection (Fig. 5c), and the Vite development server for the web front end is ready in 839 ms (Fig. 5d). These runs use the Hardhat local network and a local MongoDB instance, and local network addresses are masked.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511193



Fig. 5. Prototype running in the local development environment: (a) Hardhat node log of the DocumentRegistry deployment transaction, (b) deployment through Hardhat Ignition, (c) backend server with MongoDB connected, (d) Vite development server for the web front end. Local network addresses are masked.

V. RESULTS AND DISCUSSION

A. Functional and security testing

Test cases covering authentication, document management and sharing, IPFS and blockchain operations, document integrity and verification (TC-01 to TC-20) and security and robustness (SR-01 to SR-08) were executed against the running prototype, and the actual outcome of each was recorded. All 28 cases passed (Table VI), and the individual cases are listed in Table VII. Separately, the Hardhat test suite for the DocumentRegistry contract passed eight tests on the local network.

TABLE VI SUMMARY OF EXECUTED TEST CASES

Test group	IDs	Cases	Passed
Authentication and authorization	TC-01–05	5	5
Document management and sharing	TC-06–10	5	5
IPFS and blockchain	TC-11–15	5	5
Integrity and verification	TC-16–20	5	5
Security and robustness	SR-01–08	8	8
Total		**28**	**28**

The results support the integrity and access-control claims in the design. The same unchanged document produced identical digests (TC-16) and matched the registered hash (TC-17), whereas a modified document and an unrelated document each produced a mismatch (TC-18, TC-19, SR-03). Unauthenticated and cross-role requests were rejected, and company access without permission was denied (TC-04, TC-05, TC-20, SR-01, SR-02). Failure handling behaved as specified: simulated IPFS unavailability and blockchain transaction failure were not reported as successful (SR-05, SR-06), and modifying a signed report made signature validation fail (SR-08).

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511193

TABLE VII EXECUTED FUNCTIONAL AND SECURITY TEST CASES

ID	Scenario	Observed outcome
TC-01	Register a user with valid information	Account created; details stored in MongoDB
TC-02	Log in with valid credentials	Authenticated and redirected to the role dashboard
TC-03	Log in with an incorrect password	Login rejected with an error message
TC-04	Access a protected resource without authentication	Access denied
TC-05	Attempt an operation not permitted for the role	Operation rejected
TC-06	Institution uploads a valid academic document	Document accepted for processing
TC-07	Student uploads a supporting document	Processed by the student-document workflow
TC-08	Upload invalid or unsupported file content	Rejected with a validation error
TC-09	Student views own documents	Only permitted documents displayed
TC-10	Student shares a document with a company	Sharing recorded; authorized access provided
TC-11	Upload a valid document to IPFS	CID generated through Pinata
TC-12	Retrieve a document using its CID	Corresponding document retrieved
TC-13	Register identifier, hash and CID	Recorded in the DocumentRegistry contract
TC-14	Retrieve a registered record	Record returned by document identifier
TC-15	Invalid or duplicate registration	Handled per the contract's validation rules
TC-16	Hash an unchanged document twice	Identical SHA-256 values
TC-17	Verify an unchanged document	Hash matches the registered hash
TC-18	Modify a registered document and verify	Different hash; mismatch detected
TC-19	Submit an unrelated document against a record	Hash mismatch identified
TC-20	Request verification without permission	Request rejected
SR-01	Access without authentication	Access denied
SR-02	Access a document without sharing permission	Access denied
SR-03	Modify a registered document	SHA-256 mismatch detected
SR-04	Submit an invalid document identifier	Rejected with an error response
SR-05	Simulate IPFS unavailability	Failure reported; not shown as success
SR-06	Simulate blockchain transaction failure	Registration not reported as successful
SR-07	Access sensitive academic content without authorization	Access restricted to authorized users
SR-08	Modify a signed report and validate its signature	Signature validation failed

B. Verification case study

Table VIII compares the stages of two reports generated on 30 October 2026. In the approved case, the hash stage reported a match, the blockchain record was verified, the college record check passed and the AI stage was completed. In the rejected case, a modified marks card produced a hash mismatch, so the decision was Rejected, although the blockchain and college-record stages still passed. This is the intended behavior: the registered record is intact, and it is the submitted file that differs from it. Both reports show a valid RSA-SHA256 signature.

Each stage establishes a different property. The hash stage shows that the document under verification is identical to the registered file. The blockchain stage shows that the registered hash is the one recorded on-chain. The college-record stage links the request to a record held by the institution, and the AI stage gives the verifier visual evidence. None of these by itself shows that the institution issued a genuine document in the first place, which is why registration by the Institution is the trust anchor of the system.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511193

TABLE VIII STAGE OUTCOMES IN THE TWO REPORTS

Stage	Approved	Rejected
AI forgery detection	Completed	Completed
SHA-256 hash	Match	Failed
Blockchain record	Verified	Passed
College record	Passed	Passed
Final decision	Approved	Rejected
Signature (RSA-SHA256)	Valid	Valid

C. AI forgery detection results

A preliminary evaluation of the ML-based forgery detector was carried out on 42 documents, of which 32 were genuine marks cards and 10 were modified documents. Treating the modified (forged) class as positive, Table IX gives the confusion matrix and Table X the resulting metrics.

TABLE IX CONFUSION MATRIX OF THE ML-BASED FORGERY DETECTOR

Actual class	Predicted genuine	Predicted forged	Total
Genuine	29	3	32
Forged	1	9	10
Total	**30**	**12**	**42**

TABLE X PRELIMINARY CLASSIFICATION METRICS

Metric	Value
Accuracy	90.48%
Precision	75.00%
Recall	90.00%
F1-score	81.82%

In Table X, TP (true positives) is the number of modified documents correctly detected as forged (9), TN (true negatives) is the number of genuine documents correctly identified as genuine (29), FP (false positives) is the number of genuine documents wrongly flagged as forged (3), and FN (false negatives) is the number of modified documents missed and classified as genuine (1). Accuracy is the proportion of all documents classified correctly, (TP + TN) / (TP + TN + FP + FN) = 38/42. Precision is the proportion of flagged documents that are actually forged, TP / (TP + FP) = 9/12. Recall is the proportion of modified documents that are detected, TP / (TP + FN) = 9/10. F1-score is the harmonic mean of precision (P) and recall (R), 2PR / (P + R).

Of the 10 modified documents, 9 were detected (recall 90.00%), and 3 of the 32 genuine documents were wrongly flagged. Precision is 75.00%, so one in four flagged documents is genuine; this is tolerable because every flag is reviewed by an authorized verifier and the AI stage does not decide the outcome. The accuracy of 90.48% must be read against the class balance: a classifier that always answered “genuine” would score 76.19% (32 of 42), so precision, recall and F1-score are more informative. With only 10 modified documents, one additional missed document would lower recall to 80.00%, so these results are preliminary and should be confirmed on a larger dataset.

D. Performance

Table XI and Fig. 6 give mean times over repeated runs on academic documents. IPFS upload and blockchain confirmation dominate, at 3.5 s and 4.0 s, while hashing and hash comparison take tens of milliseconds, which supports keeping the heavy data off-chain and the on-chain record small. IPFS retrieval took 2.0 s and report generation 1.0 s. The AI analysis stage took 1.5 s on average over 10 runs.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511193

TABLE XI MEAN EXECUTION TIME PER OPERATION

Operation	Runs	Mean time
SHA-256 hash generation	10	15 ms
IPFS upload	5	3.5 s
IPFS retrieval	5	2.0 s
Blockchain registration	5	4.0 s
Hash verification	10	20 ms
AI analysis stage	10	1.5 s
Report generation	5	1.0 s

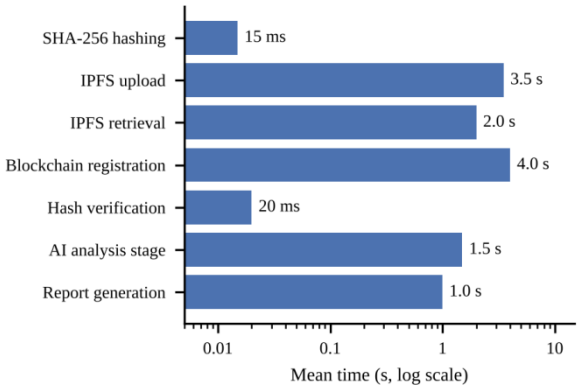

Fig. 6 Mean time per operation (logarithmic axis).

The documents were within the 5 MB upload limit. Only means were recorded, so variability is not reported. These figures are not directly comparable with those of other systems [7] because file sizes, networks and hardware differ. Registration is the slowest workflow, since it combines hashing, IPFS upload and a blockchain transaction, but it is performed once per document by the institution. Verification involves only retrieval, hashing and comparison, and so is faster for the end user.

E. Comparison with related systems

Table XII compares SecureChain qualitatively with selected systems from the survey. An entry "—" means the feature is not reported in the cited work.

TABLE XII QUALITATIVE COMPARISON

System	Platform / storage	Access control	AI analysis	Signed report
[7]	Ethereum Sepolia; on-chain hash	—	—	—
[6]	Ethereum; IPFS (Pinata); MySQL	Hash-only sharing	—	—
[5]	Ethereum; IPFS	Role-based, approval	—	—
[2]	Private chain (PoS); private IPFS	Token-based sharing	—	—
[13]	Polygon; IPFS	Four-party model	—	—
[16]	No blockchain	—	CNN	—
Secure Chain	**Ethereum Sepolia; IPFS (Pinata); MongoDB**	**Four roles, verifier approval**	**Prelim.**	**RSA-SHA256**



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511193

F. Discussion

The results show that the main design idea works at prototype scale: a registered document can be re-checked in milliseconds against an on-chain record, and a modified copy is caught without contacting the institution. Three observations matter for practice. First, the stage structure separates different failure causes. In the rejected case the registered record was intact and only the submitted file differed, which is information a single pass/fail check would hide. Second, the human verifier remains the decision maker, which suits a setting where an error has consequences for a student's career, and the signed report gives the company evidence that can be checked later. Third, the cost profile favors the design: the expensive operations occur at registration, while repeated verification is cheap.

The AI-based document analysis component is designed to provide additional evidence during the verification process by examining document characteristics using Machine Learning techniques. Unlike cryptographic hash comparison, which identifies changes in the registered file, AI-based analysis can assist in identifying potential visual or structural irregularities. The analysis is intended to support the authorized faculty verifier rather than independently determine document authenticity. A preliminary evaluation of its detection capability is reported in Section V-C; it uses a small labelled dataset, so the figures indicate feasibility and are not conclusive.

Table XIII relates each design goal to its mechanism and to the evidence gathered.

TABLE XIII DESIGN GOALS AND SUPPORTING EVIDENCE

Goal	Mechanism	Evidence
Integrity	SHA-256 digest compared with the on-chain hash	TC-16 to TC-19, SR-03
Confidentiality	AES-256 encryption of institutional documents	SR-07
Access control	JWT, role-based permissions, explicit provision	TC-04, TC-05, TC-20, SR-01, SR-02
Accountability	Verifier decision; RSA-SHA256 signed report	SR-08, Table VIII
Honest failures	Failures reported, not shown as success	SR-05, SR-06

G. Privacy considerations

Academic documents contain personal data. In SecureChain a company reaches a student's document only through the provision workflow, institutional documents are stored encrypted, and the blockchain holds only the identifier, hash and CID, so no document content is published on-chain.

H. Limitations

The results should be read with the following limits in mind.

- **AI module:** the evaluation is preliminary and uses a small dataset (32 genuine and 10 modified documents) from one institution; precision is 75.00%, so some genuine documents are flagged, and the results need confirmation on a larger dataset.
- **Scale:** tests were run by the development team on a single machine with small numbers of documents and runs; production load, many concurrent users and Ethereum Mainnet costs were not evaluated.
- **Server-side hashing:** the backend sees the plaintext document before encryption, unlike client-side designs [6], [7].
- **Encryption:** AES-CBC provides confidentiality but no authentication of the ciphertext, and a single key held in environment configuration is a weak point.
- **Scope of the guarantee:** a hash match shows that a file equals the registered file, not that the issuer was genuine; trust still rests on the institution's registration.
- **Storage privacy:** IPFS request metadata can leak access patterns [17].
- **Test environment:** results come from a test network and a local database; failure handling was simulated rather than observed in production.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511193

VI. CONCLUSION AND FUTURE WORK

SecureChain combines blockchain registration, IPFS storage, AES-256 encryption, SHA-256 integrity checking, role-based access control and signed verifier reports in one prototype. All 20 functional and 8 security test cases passed, a modified document was rejected through a hash mismatch, and the main cost of registration came from IPFS upload and blockchain confirmation rather than hashing. A preliminary evaluation of the ML-based forgery detector on 42 documents reached 90.48% accuracy and an 81.82% F1-score.

Several extensions follow from the limitations above:

- **AI evaluation:** extend the evaluation with more modified documents (for example, edited marks or changed names and numbers), compare the Random Forest and SVM classifiers with student-grouped splits, and report variance across folds in addition to the confusion matrix, precision, recall and F1-score.
- **Client-side hashing:** compute the digest in the browser so that the server never handles unencrypted content [6], [7].
- **Stronger encryption:** use authenticated encryption such as AES-GCM with managed keys instead of a key in environment configuration.
- **Lower cost:** batch registrations with Merkle trees and deploy on a Layer-2 network [11], [15].
- **Privacy:** add selective disclosure so that a company learns only the facts it needs [14], [15].
- **Larger evaluation:** test with more users, concurrent load and a public network, and report variance as well as means.

REFERENCES

[1]. N. Nizamuddin, K. Salah, M. A. Azad, J. Arshad, and M. H. Rehman, “Decentralized document version control using ethereum blockchain and IPFS,” Computers and Electrical Engineering, vol. 76, pp. 183–197, 2019.
[2]. A. K. Shrivastava, C. Vashistth, A. Rajak, and A. K. Tripathi, “A Decentralized Way to Store and Authenticate Educational Documents on Private Blockchain,” in 2019 2nd International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Ghaziabad, India, 2019, pp. 1–6.
[3]. S. K. Ambast and T. A. Sumesh, “A Blockchain Based Credential Verification System using IPFS,” in 2022 IEEE 19th India Council International Conference (INDICON), Kozhikode, India, 2022, pp. 1–7.
[4]. A. Chowdhary, S. Agrawal, and B. Rudra, “Blockchain based Framework for Student Identity and Educational Certificate Verification,” in 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), Surathkal, India, 2021, pp. 916–921.
[5]. T. Rahman, S. I. Mouno, A. M. Raatul, A. K. A. Azad, and N. Mansoor, “Verifi-Chain: A Credentials Verifier using Blockchain and IPFS,” arXiv preprint arXiv:2307.05797, 2023.
[6]. S. Kadwe, Y. Patle, S. Laddha, A. Ghule, and Y. Patil, “EduDocs: Document Verification using Blockchain,” in 2024 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS), Pune, India, 2024, pp. 1–5.
[7]. M. M. Islam, A. S. Nafi, K. R. Ullah, J. Hossain, A. F. H. Dhrubo, S. Pramanik, and M. A. Qayum, “Blockchain-Based Document Verification System Using Ethereum Smart Contracts and Cryptographic Hashing,” in 2026 IEEE 2nd International Conference on Secure IoT, Assured and Trusted Computing (SATC), Dhaka, Bangladesh, 2026, pp. 1–5.
[8]. E. Nyaletey, R. M. Parizi, Q. Zhang, and K.-K. R. Choo, “BlockIPFS - Blockchain-enabled Interplanetary File System for Forensic and Trusted Data Traceability,” in 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 2019, pp. 17–24.
[9]. X. Zhai, S. Pang, M. Wang, S. Qiao, and Z. Lv, “TVS: a trusted verification scheme for office documents based on blockchain,” Complex & Intelligent Systems, vol. 9, pp. 2865–2877, 2023.
[10]. M. Aldwairi, M. Badra, and R. Borghol, “DocCert: Nostrification, Document Verification and Authenticity Blockchain Solution,” in Proceedings of the IEEE Conference on Blockchain Applications, Irbid/Dubai, 2023, pp. 1–8.
[11]. I. Meirobie, A. P. Irawan, H. T. Sukmana, D. P. Lazirkha, and N. P. L. Santoso, “Framework authentication e-document using blockchain technology on the government system,” International Journal of Artificial Intelligence Research, vol. 6, no. 2, 2022.
[12]. S. Sarang, D. Rana, S. Patel, D. Savaliya, U. P. Rao, and A. Chaurasia, “Document Management System Empowered by Effective Amalgam of Blockchain and IPFS,” Procedia Computer Science, vol. 215, pp. 340–349, 2022.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511193

[13]. K. Goswami, D. Vaithiyanathan, P. Verma, and B. Kaur, "Document verification using Blockchain," in 2024 International Conference on Advances in Modern Age Technologies for Health and Engineering Science (AMATHE), Delhi, India, 2024, pp. 1–6.
[14]. A. Satybaldy, A. Subedi, and M. Nowostawski, "A Framework for Online Document Verification Using Self-Sovereign Identity Technology," Sensors, vol. 22, no. 21, p. 8408, 2022.
[15]. J. A. Berrios Moya, J. Ayoade, and M. A. Uddin, "A Zero-Knowledge Proof-Enabled Blockchain-Based Academic Record Verification System," Sensors, vol. 25, no. 11, p. 3450, May 2025.
[16]. K. Maamouli, A. Cheddad, H. Benhamza, and A. Djeffal, "A CNN based architecture for forgery detection in administrative documents," in 2022 International Symposium on iNnovative Informatics of Biskra (ISNIB), Biskra, Algeria, 2022, pp. 1–6.
[17]. L. Balduf, S. Henningsen, M. Florian, S. Rust, and B. Scheuermann, "Monitoring Data Requests in Decentralized Data Storage Systems: A Case Study of IPFS," arXiv preprint arXiv:2104.09202, 2022.

 This work is licensed under a Creative Commons Attribution 4.0 International License