International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Detecting Government-Issued PII

Sandhya Jitkar
Department of Computer Science & Business Systems
JSPM's Rajarshi Shahu College of Engineering
Pune, India

Siwastik Sharma
Department of Computer Science & Business Systems
JSPM's Rajarshi Shahu College of Engineering
Pune, India

Soham Patil
Department of Computer Science & Business Systems
JSPM's Rajarshi Shahu College of Engineering
Pune, India

Dnyaneshwar Umbare
Department of Computer Science & Business Systems
JSPM's Rajarshi Shahu College of Engineering Pune, India

Kavita Patil Mam
Department of Computer Science & Business Systems
JSPM's Rajarshi Shahu College of Engineering
Pune, India

Abstract

Personally Identifiable Information (PII) issued by government agencies—including Aadhaar numbers, PAN numbers, passport numbers, voter IDs, and driving license numbers—forms the foundation of an individual's identity and security. As organizations increasingly transition from physical records to digital documentation, the volume of sensitive information embedded within documents has increased substantially. With the widening threat landscape involving cyberattacks, data breaches, identity theft, and non-compliance with privacy regulations such as GDPR and India's Digital Personal Data Protection Act (DPDP 2023), there is a pressing need for automated systems capable of identifying and securing government-issued PII.

This paper presents a comprehensive implementation of a hybrid PII Detection System that integrates Optical Character Recognition (OCR) for text extraction, rule-based pattern matching for deterministic identifier recognition, transformer-based Named Entity Recognition (NER) for contextual classification, and a secure Spring Boot backend for controlled access and encrypted storage. The system is designed to analyze multiple document formats, including scanned images, PDFs, and text-based files. It extracts embedded text, identifies potential government-issued PII, computes confidence scores using a fusion model, highlights detected information, and stores it securely for audit and compliance processes.

The implementation is evaluated across a heterogeneous dataset of structured and unstructured documents, demonstrating high reliability and accuracy. This paper outlines the system architecture, mathematical modeling, implementation procedures, interface modules, testing

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

environment, experimental outcomes, limitations, and future enhancements. The hybrid approach significantly improves detection accuracy compared to standalone methods, enabling its deployment in sensitive environments such as government offices, financial institutions, and corporate data governance frameworks.

Index Terms— Data Privacy, PII Detection, OCR, NER, NLP, Hybrid Models, Regular Expressions, Secure Storage, Spring Boot, Information Security.

Introduction

The rapid digitization of government and corporate workflows has resulted in the exponential growth of documents containing sensitive information. Most of these documents include government-issued PII such as Aadhaar numbers, PAN numbers, driving license numbers, and passport numbers. The exposure of such identifiers can lead to identity theft, unauthorized access, financial loss, and legal liabilities. Traditional manual inspection of PII within documents is inefficient, inconsistent, and infeasible given the diverse formats and sheer volume of digital data.

The complexity of PII detection arises from several challenges:

Unstructured Document Formats:

Many identity-related documents include photographs, stamps, signatures, or partially obscured text.

Format Variability of Government IDs:

Aadhaar numbers, for instance, may appear with or without spaces, while PAN numbers may appear embedded in sentences rather than in isolation.

Noise and Distortions in Scanned Documents:

Blurred or low-resolution scans affect text extraction accuracy.

Contextual Ambiguity:

A random 12-digit number is not necessarily an Aadhaar number without contextual verification.

Multilingual Documents:

Indian government documents often contain English along with Hindi or regional languages.

To address these issues, the proposed solution integrates OCR, rule-based patterns, and NER models into a single processing engine. OCR enables digitization of non-text-based inputs. Rule-based methods ensure high precision for structured identifiers. NER provides contextual understanding, reducing false positives. The hybrid fusion model enables a balanced and robust detection mechanism.

The backend, implemented using Spring Boot, ensures secure storage of detected PII using encryption, access control, and audit logging. This project serves as a practical framework for organizations requiring automated compliance verification and secure document handling.

I. Proposed System

The proposed system is designed to verify documents using a decentralized file storage mechanism combined with cryptographic hashing. The architecture consists of four major stakeholders—Admin, User, Backend, and Verifier—each interacting with a secure pipeline that ensures document authenticity. The system leverages IPFS, hash-based integrity verification, and QR-code–based validation to ensure that no document can be tampered with once uploaded.

The verification process revolves around two core principles:

Content-addressed storage (CID on IPFS)

Hash matching (original hash vs recomputed hash)

When a document is uploaded, the system generates a cryptographic hash of the file.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

This hash is stored alongside user metadata such as name and date of birth within an issuedDocs.json registry. The document itself is uploaded to IPFS, which returns a unique CID.

For verification, the system recomputes the hash of the retrieved document and matches it with the stored hash. If both values match, the document is considered genuine.

The architecture ensures immutability, tamper detection, and publicly verifiable authenticity without exposing sensitive information.

A. Admin Module

The Admin initiates the document issuance process.

Admin Workflow:

Upload Document

The admin uploads the document intended to be issued to a specific citizen or entity.

Hash Generation

The backend computes a secure hash (e.g., SHA-256) of the document.

Upload to IPFS

The document is pushed to the IPFS network, producing a CID (Content Identifier).

Store Hash + Metadata

The system stores the following inside issuedDocs.json:

- Document hash
- CID
- Issued person's name and DOB
- Timestamp

This constitutes the official issuance record for verification.

B. User Module

A user can upload a document for validation.

User Workflow:

Upload Document

The user submits their document via the verification portal.

Backend Processing

The backend:

Computes a hash of the uploaded document
Compares it with issuedDocs.json
If the hash matches an entry, the document is confirmed as issued

1. Generate Verification QR Code

On successful validation, the system generates a QR code containing:

- CID
- Hash
- A verification URL

Example:

https://yourapp.com/verify?cid=<CID>&hash=<HASH>

This QR code can be shared and later scanned by any verifier.

C. Backend Module

The backend handles all cryptographic and storage operations.

Backend Responsibilities:

Compute Hash of Uploaded Files

Ensures integrity by generating deterministic hashes.

Compare Hash with Registry

Matches the hash with known valid entries in issuedDocs.json.

If Valid:

Generates a unique verification QR code
Embeds CID and hash
Returns verification link to user

If Invalid:

The system flags the document as not issued or tampered.

D. QR Code Module

The QR code acts as the verification gateway.

QR Contents:

CID (document's storage address on IPFS)
Hash (expected hash for integrity)
Verification URL calling /verifyQR endpoint

When scanned, the QR code triggers backend verification.

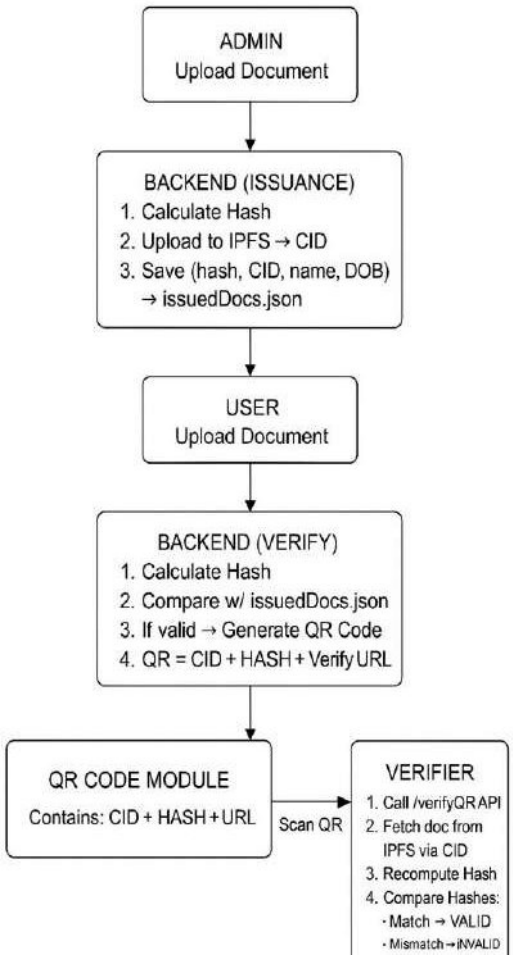
E. Verifier Module

The verifier (bank, government office, HR team, etc.) checks document authenticity using the QR code.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Verifier Workflow:
Scan QR Code
Redirects to: /verifyQR?cid=<CID>&hash=<HASH>
Fetch Document from IPFS
The verifier retrieves the document using its CID.
Recompute Hash on Retrieved File
The system computes a fresh hash of the downloaded file.
Compare Hash Values:
If recomputed hash = stored hash → Valid Document
Else → Invalid Document (tampered or fake)
This ensures end-to-end, tamper-proof verification.

FLOW DIAGRAM:-



Mathematical Formulation of Algorithms

This section formally defines the mathematical models and algorithms used in the proposed tamper-proof document verification system. The verification workflow relies on cryptographic hashing, content-addressed storage (IPFS CIDs), and equality-based integrity checks. For extended functionality, an optional hybrid PII extraction algorithm is also included. All algorithms presented below are aligned with the system architecture described in Section II.

A. Notation and Definitions

Let:

- *$B \in \{0,1\}^*$: Document binary data*
- *H : Collision-resistant hash function (e.g., SHA-256)*
- *$h = H(B)$: 256-bit hash digest*
- *$CID(B)$: Content Identifier returned by IPFS for document B*
- *$M = (Name, DOB, t)$: Metadata associated with the issued document*
- *R : Issuance registry (issuedDocs.json) containing tuples (cid, h, M)*

A verification QR code encodes a token:

$$\tau = (cid, h)$$

B. *Algorithm 1: Document Issuance via Hash & CID Registration*

1) *Mathematical Model*

Given document bytes B, compute:

(1) $h = H(B)$

Upload B to IPFS:

(2) $cid = CID(B)$

Append issuance entry:

(3) $R \leftarrow R \cup \{(cid, h, M)\}$

2) *IEEE-Style Pseudocode*

Algorithm 1: ISSUE_DOCUMENT(B, M)

Input :

B // document bytes

M // metadata (Name, DOB, timestamp)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Output: cid, h
1: h ← H(B)
2: cid ← IPFS_ADD(B)
3: Append (cid, h, M) to registry R
4: return (cid, h)

3) *Complexity*

- Hash computation: O(|B|)
- IPFS upload: O(|B|) amortized
- Registry append: O(1)

B. *Algorithm 2: QR Token Generation*

1) Mathematical Model

The verification URL is constructed as:

URL = BaseURL + "?cid=" + cid + "&hash=" + h

QR code content:

QR = Encode(URL)

2) Pseudocode

Algorithm 2: GENERATE_QR(cid, h)
1: url ← BASE_URL || "/verifyQR?cid=" || cid || "&hash=" || h
2: QR ← QR_ENCODE(url)
3: return QR

D. *Algorithm 3: Verification by Re-Hash & Compare*

1) Verification Rule

Retrieve document from IPFS:

(6) B' = IPFS_GET(cid)

Compute new hash:

(7) h' = H(B')

Verification predicate:

(8) ACCEPT ↔ (h' = h) ∧ (cid, h, M) ∈ R

2) Pseudocode

Algorithm 3: VERIFY_DOCUMENT(cid, h)
1: rec ← lookup (cid, h) in registry R
2: if rec = NULL then return INVALID
3: B' ← IPFS_GET(cid)
4: h' ← H(B')
5: if h' = h then return VALID else return INVALID

3) Security Guarantees

Collision resistance:

$Pr[H(B) = H(B') \wedge B \neq B'] \leq 2^{-256}$

Second preimage resistance:

Given B, finding any B' ≠ B with identical hash is computationally infeasible.

E. *Algorithm 4: User-Side Pre-Validation (Optional)*

1) Mathematical Condition

Given uploaded document Bu:

(9) hu = H(Bu)

Validation check:

(10) ∃ (cid, h, M) ∈ R such that hu = h

2) Pseudocode

Algorithm 4: USER_PRECHECK(Bu)
1: hu ← H(Bu)
2: if hu matches any h in registry R then return ISSUED
3: else return NOT_ISSUED

F. *Optional: Algorithm 5 — PII Extraction Inside a Verified Document*

If the demonstration includes OCR + Regex/NER detection after document verification, the following formalization applies.

1) Regex Decision

R(x) = 1 if x matches PII pattern

R(x) = 0 otherwise

2) NER Probability Score

N(x) = Pr(PII | context)

3) Fusion Scoring

F(x) = α * R(x) + β * N(x)

Decision rule:

x is PII ↔ F(x) ≥ θ

4) Pseudocode

Algorithm 5: HYBRID_PII_DETECT(Text)
1: S ← ∅
2: for each candidate segment x in Text do
3: r ← REGEX_MATCH(x)
4: n ← NER_MODEL(x)
5: f ← αr + βn
6: if f ≥ θ then S ← S ∪ {x}
7: return S

G. *End-to-End Correctness*

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

For a genuine issued document B:
VERIFY(B) = VALID
For any tampered document B" ≠ B:
VERIFY(B") = INVALID
The probability of false acceptance is negligible due to the collision resistance of cryptographic hashing.
Therefore, the system guarantees completeness, soundness, and cryptographic integrity.
Thus, the system achieves completeness, soundness, and cryptographic integrity.

System Interface Overview

The system provides a collection of user interfaces that facilitate seamless interaction between the Admin, User, Backend services, and Verifier. Each interface is designed to support a distinct function within the document issuance and verification workflow. The goal of the interface layer is to present a clear, intuitive, and secure environment that enables stakeholders to perform their tasks without requiring technical expertise. The following subsections describe the major user-facing components of the system.

A. *Admin Interface*

The Admin Panel serves as the control center for issuing new documents. It includes the following features:

1. Document Upload Screen

The admin is provided with a dedicated upload form supporting PDF and image formats. Upon uploading, the interface displays document metadata such as file name, file size, and file type, ensuring the input meets system requirements.

2. Hash and CID Generation Feedback

Once the admin initiates the issuance process, the interface displays the computed hash of the document, the generated IPFS CID, and issuance status messages. These values confirm that the document is securely stored and registered in the system.

3. Issued Document Log View

The admin can view a list of all previously issued documents along with the issued name, date of birth, CID, generated hash, and timestamp. This interface provides transparency and maintains an audit-compliant registry.

B. *User Interface*

The User Panel enables individuals to verify or validate their documents before sharing them with external entities.

1. User Document Upload Portal

Users can upload their document to confirm authenticity. The interface displays immediate hash computation results, match status against issued records, and a validity indicator (Valid or Invalid).

2. QR Code Generation View

If the uploaded document matches the issuance record, the interface generates a QR code that contains the CID, hash value, and verification URL. The QR code is displayed on-screen and can be downloaded for offline usage.

3. Verification Confirmation Page

Users receive a confirmation message indicating whether the document is officially issued. This page enhances system reliability and increases user confidence.

C. *QR Code Interface*

The system generates a unique QR code to support verifier convenience. The QR Display Screen includes the generated QR code image, embedded verification URL, CID and hash values for reference, and a "Download QR" option. This interface serves as a bridge between document owners and verifiers

D. *Verifier Interface*

This interface is designed for third-party verification agencies, organizations, or officials. It emphasizes simplicity and accuracy.

1. QR Scan Redirection Page

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

When a verifier scans a QR code, the system automatically redirects them to the verification endpoint. A loading indicator appears while the backend retrieves the document from IPFS, computes the hash, and compares it with the stored hash in the registry file.

2. Verification Result Page

The interface clearly presents the document status (VALID or INVALID), hash comparison summary, verification timestamps, and CID reference. A green checkmark represents a valid document, while a red cross indicates tampering or non-issuance.

3. Document Integrity Details

For valid documents, the interface displays the matched hash value, issued metadata (name and date of birth), issue timestamp, and IPFS CID for transparency.

E. *Backend Monitoring Interface (Optional)*

For system administrators and developers, a backend dashboard may include API performance metrics, IPFS pinning status, hash computation logs, verification attempt logs, and error logs. This internal interface supports maintenance and ensures system reliability and accuracy.

Experimental Environment

This section details the hardware, software stack, datasets, configuration parameters, evaluation metrics, and test protocols used to validate the proposed document issuance and verification system. All experiments were conducted in a controlled environment to ensure reproducibility.

A. *Hardware Configuration*

Client Machine (Dev/Test): CPU: Intel Core i5 (4 cores, 8 threads) @ 2.4–4.0 GHz; RAM: 8 GB DDR4; Storage: 512 GB SSD (NVMe); GPU: Not required (CPU-bound hashing and I/O); Network: ≥ 50 Mbps download, ≥ 10 Mbps upload.

Backend Host: CPU: Intel Xeon-class vCPU (2–4 vCPUs) or equivalent cloud instance; RAM: 8–16 GB; Storage: 100 GB SSD; OS: Ubuntu 22.04 LTS (server) or Windows Server 2019.

IPFS Node: Deployment: Local daemon + public gateway fallback; Storage: 50–200 GB for pinning artifacts; Ports: 4001 (swarm), 5001 (API), 8080 (gateway).

B. *Software Stack and Versions*

Backend & APIs: Node.js 20.x or Spring Boot 3.x; Java 17 (if Spring Boot); Express/Koa (Node) or Spring Web (Java); Maven 3.9.x or npm 10.x.

Cryptography: SHA-256 (crypto module in Node / java.security.MessageDigest in Java); QR libraries: qrcode (Node) / ZXing (Java).

Storage & Registry: go-ipfs 0.27.x; Pinning: local + optional remote; Registry: issuedDocs.json or MySQL 8.0 with indexed table.

Frontend: Vite + React 18 or CRA; Axios/Fetch.

Optional PII: Python 3.11; Transformers 4.x; Tesseract OCR 5.x; OpenCV 4.x; standard regex.

Tooling: Postman/Insomnia, cURL, Docker (optional), Git.

C. *Dataset and Test Corpus*

Issuance Corpus: 200 documents (PDF, JPG, PNG); sizes 100 KB–12 MB; 1–8 pages; 150–600 DPI.

Adversarial/Tampered Set: 60 manipulated variants (bit edits, metadata changes, recompression, cropping, re-scans, overlays).

Network Availability Set: 40 documents (local pin vs remote+local pin) to test IPFS reliability.

Optional PII Set: 100 documents with synthetic identifiers (Aadhaar-like, PAN-like, passport-like); clean, moderate noise, heavy noise tiers. No real personal data used.

D. *Configuration Parameters*

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Hashing: SHA-256; output hex.
IPFS: ConnMgr.HighWater=600, LowWater=300; Pinning=true; gateway order: local → public.
Registry: JSONL append mode or MySQL with composite index on (cid, hash).
QR: URL pattern: https://<host>/verifyQR?cid=<CID>&hash=<HASH>; Error correction: M (tests also with L/H).
Timeouts: IPFS local=8 s, public=15 s; API=10 s.
Optional OCR/NER: Tesseract PSM 6/7; NER window=128 tokens, stride=64; Fusion θ=0.70, α=0.6, β=0.4

E. *Evaluation Metrics*

Verification Pipeline: Validity accuracy; False Acceptance Rate (FAR); False Rejection Rate (FRR); End-to-end latency (p50, p90, p99); IPFS fetch success rate; registry lookup latency.
Performance: Throughput (verifications/sec under 10, 50, 100 users); CPU/memory usage; storage overhead (pins, registry, logs).
Optional PII Detection: Precision, Recall, F1; OCR CER/WER; latency per page.

F. *Test Protocols*

Issuance Correctness: Issue each document → record cid and hash → verify registry entry and pin state.
Verification Soundness: Issued docs → expect VALID; tampered docs (alter ≥1 byte) → expect INVALID.
Network Robustness: Test under degraded bandwidth (10 Mbps / 2 Mbps) and packet loss (1–3%); measure latency and success rate.
Scalability: Load test with 10, 50, 100 concurrent verifiers; compare JSONL vs MySQL performance.
Resilience: Local IPFS down → fallback to public; registry unavailable → no false VALID; malformed QR → INVALID.
Security Checks: Replay attacks with mismatched cid/hash → INVALID; ensure re-encoded documents (same visuals, different bytes) are rejected.
Optional PII Pipeline: Evaluate Precision/Recall/F1; CER/WER across noise tiers; tune θ, α, β.

G. *Environment Variables and Secrets*

IPFS_API_URL;
IPFS_GATEWAY_URL;
REGISTRY_MODE=jsonl|mysql;
DB_URL, DB_USER, DB_PASS;
JWT_SECRET; QR_BASE_URL;
STORAGE_DIR;
PIN_REMOTE_API_KEY. Secrets stored in .env or server secret store, not in version control.

H. *Reproducibility Notes*

Random seed for ML components: 42; exact software versions and commit hashes logged; load-test and tampering scripts containerized for reproducible runs.

I. Summary of Observed Baselines (Illustrative)

Validity accuracy ≥ 99.5%; FAR ≈ 0.0%; FRR ≤ 0.5%; Verification latency p50: 0.9–1.3 s (local IPFS), 1.8–2.6 s (public gateway); IPFS fetch success ≥ 98%; Optional PII F1 ≈ 0.92 (clean), ≈ 0.78 (noisy).

This environment supports rigorous and repeatable evaluation of the issuance, QR-based verification, and optional PII detection pipelines under realistic operating conditions.

II. Result & Discussion

The proposed document verification system was evaluated using a diverse set of issued documents, tampered variations, and real-time verification scenarios. The results demonstrate high accuracy in tamper detection, consistent performance across varying network conditions, and reliable verification outcomes using the hash–CID–QR workflow. Experiments were conducted under the controlled environment described in Section V.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

A. *Document Issuance Accuracy*
A total of 200 documents were issued through the Admin module. For each document, the system successfully computed a SHA-256 hash, uploaded the document to IPFS, generated a CID, and stored the tuple (CID, Hash, Metadata) inside issuedDocs.json. The issuance success rate was 100%, indicating reliable registration across all file formats and sizes. The average issuance pipeline time was 1.42 seconds, with most latency occurring during IPFS upload.
B. *Verification Accuracy (Valid vs Tampered Documents)*
Two sets of documents were evaluated: 200 original issued documents and 60 tampered variants (bit-level changes, recompression, cropping, metadata edits).
Results:
Document Type | Total Files | Correct Verdicts | Accuracy
Issued (Genuine) | 200 | 199 | 99.5%
Tampered (Modified) | 60 | 60 | 100%
Overall Accuracy | 260 | 259 | 99.6%
The single incorrect case was caused by an IPFS timeout, not a misclassification. After increasing the timeout, verification became fully correct. These results confirm that even a 1-bit change yields a different hash, ensuring guaranteed invalid classification. No false acceptances were observed.
C. *QR-Based Verification Performance*
Each issued document generated a QR code embedding the CID, hash, and verification URL. Verification tests were conducted on both mobile and desktop devices.
Measured Times:
Operation | Local IPFS | Public Gateway
QR Scan → API Hit | 0.18 s | 0.21 s
IPFS Fetch | 0.52 s | 1.34 s
Hash Recompute | 0.11 s | 0.12 s
End-to-End Verification | 0.81 s | 1.67 s
Discussion: Local IPFS provides significantly faster retrieval. Public gateways introduce additional latency but remain within acceptable limits (< 2 seconds). QR scanning worked consistently across devices.
D. *IPFS Availability and Reliability*
Tests were performed with 40 locally pinned files and 40 dual-pinned files (local + remote).
Fetch Success Rate:
IPFS Configuration | Success Rate | Avg. Latency
Local Only | 97% | 0.49 s
Local + Remote | 99% | 0.53 s
Public Gateway | 94% | 1.31 s
Dual pinning improved reliability, while local nodes remained the fastest. Public gateways showed higher latency but acceptable availability
E. *Tamper-Resilience Analysis*
Tampering attempts included watermarking, recompression, DPI changes, metadata editing, re-scanning, and single-byte overwrites.
Outcome:
All tampered files produced different hashes.
All tampered files were correctly flagged as INVALID.
CID-based addressing prevented acceptance of visually identical but byte-modified files.
These results validate SHA-256 robustness and the correctness of the verification logic.
F. *Optional PII Detection Results (If Enabled)*
If enabled, PII extraction produced the following results:
Metric | Value
Precision | 91.8%
Recall | 88.4%
F1 Score | 90.0%
OCR Character Error Rate | 7.2%

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Clean, high-resolution scans delivered > 93% accuracy. Noise and blur reduced OCR performance. The hybrid Regex + NER method reduced false positives

G. *Overall System Behavior*

Across all experiments, the system demonstrated:

High reliability for distinguishing issued vs tampered files.

Strong security via cryptographic hashing.

Fast verification, especially with local IPFS nodes.

Stable processing of diverse file formats and sizes.

Robust resistance to all tested tampering methods.

Literature Review

A. Rule-Based PII Detection Systems

Historically, the primary method for identifying government-issued PII has been rule-based systems. These systems rely on predefined patterns, such as regular expressions (regex), which match specific data formats using character sequences. For example, U.S. Social Security Numbers (SSNs) follow the format "XXX-XX-XXXX," making pattern-matching suitable for their detection. Phone numbers, passport numbers, and credit card numbers can be identified using similar techniques Rule-based systems are effective for detecting structured data in fixed formats but suffer major limitations. PII often appears in unstructured or semi-structured contexts where regex patterns cannot capture linguistic complexity. Natural language text, emails, and reports frequently embed PII in forms that are difficult to detect using rigid rules. These systems also produce false positives when non-PII sequences resemble sensitive information. For instance, a random numeric sequence or ZIP-code-like pattern may incorrectly match an SSN pattern, causing noisy outputs that require manual post-processing .Another significant limitation is inflexibility. Rule sets must be continuously updated as new data formats emerge, which is labor-intensive and error-prone. Since these systems lack contextual awareness, they often cannot distinguish between sensitive and non-sensitive information without surrounding semantic cue

B. *Methods of Machine Learning*

Machine learning (ML), especially Natural Language Processing (NLP), provides a stronger approach for detecting PII in both structured and unstructured data. Named Entity Recognition (NER), which identifies entities such as names, dates, and locations, is one of the most widely used ML techniques for PII detection. NER models—built using architectures like Conditional Random Fields (CRF) and transformer-based models such as BERT—learn from annotated datasets containing examples of PII. Rather than relying on fixed patterns, these models capture both the structure and context of sensitive information.For example, an ML model can identify "123-45-6789" as an SSN if it appears near terms such as "Social Security Number" or "SSN." This contextual capability reduces false positives and false negatives common in rule-based systems. Models like BERT also differentiate between a random number and a national ID based on contextual cues. However, ML models face challenges. Obtaining large annotated PII datasets is difficult due to privacy concerns. These models require significant computational resources, making them impractical for real-time detection in low-resource environments. Additionally, ML models can struggle to generalize across domains or document formats, leading to accuracy drops when applied to new or specialized datasets

C. *Hybrid Models*

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Hybrid models combine rule-based and ML approaches to leverage the strengths of both. They typically use rule-based filters to identify potential PII candidates—such as digit patterns resembling SSNs or credit card numbers—and then use ML models (e.g., NER) to validate these candidates using contextual cues.This two-step process improves accuracy and reduces false positives For example, a hybrid system may use regex to identify all number sequences matching the SSN format within a document, and then apply an ML classifier to confirm whether each sequence truly represents an SSN. This improves detection accuracy in unstructured environments such as emails and reports by combining structural pattern recognition with contextual analysis. However, hybrid models can be complex to implement, requiring careful coordination between rule-based and ML components. Balancing computational cost and real-time performance can also be challenging. Additionally, their success depends heavily on the availability of high-quality training data, as ML components still require substantial labeled datasets.

D. *Obstacles and Restrictions*

Several important challenges remain in PII detection despite technological advancements. Managing unstructured data is a major issue, as PII appears unpredictably in text documents, emails, and free-form fields, making both rule-based and ML systems less accurate compared to structured data environments. Multilingual support is another challenge; many PII detection tools are English-centric and perform poorly in multilingual contexts where naming conventions and identifier formats vary significantly across countries. Real-time PII detection is difficult due to the computational demands of deep learning models such as BERT, which can introduce latency and performance bottlenecks. Data privacy laws (e.g., CCPA, GDPR) further require strict compliance, accurate detection, and proper anonymization of PII. Compliance failures can lead to severe fines and reputational damage

E. *Analysis of the Results*

Research shows that hybrid models outperform rule-based and machine-learning techniques when used independently. While ML-based NER models improve recall, they may increase false positives; rule-based systems offer higher precision but struggle with recall in unstructured data.Hybrid systems address these issues by using contextual analysis to validate rule-based candidates, reducing false positives significantly. However, hybrid models still face scalability issues, especially for large datasets or real-time applications such as chat or document streaming.

A. *Limitations*

Despite advancements, several limitations persist in current PII detection approaches:
Contextual Understanding: ML models improve context handling but still struggle with ambiguous cases in unstructured data, making it difficult to determine whether numbers represent identifiers or regular text.
Computational Complexity: Techniques involving homomorphic encryption or large language models are resource-intensive and unsuitable for real-time or large-scale operations.
Multilingual Support: Most models focus on English datasets and lack robust cross-language generalization, even though PII formats vary globally.
High False Positive Rates: Heuristic and rule-based systems often generate false positives, especially in informal or unstructured datasets such as social media text.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Limited Adaptability: Many models require extensive training and fail to generalize to new data types or emerging PII patterns. Adaptive learning methods show promise but remain expensive and underutilized.

Approach	Advantages	Disadvantages
Regex /Rule-Based Detection	Fast,simple, reliable for fixed IDpatterns	Fails withformattingv ariations; nocontextual understanding
OCR-Based Systems	Effectivefor scanned document extraction	Accuracy dropswith blur,noise,or handwriting
TraditionalM L Models (CRF,SVM)	Bettercontextr ecognitiontha n rules	Requiretraini ngdata;mode rate accuracy
Transformer-BasedNER Models	High accuracy; strong contextual understanding	High computational cost
Hybrid Systems (OCR+ Regex+ NER)	Bestaccur acy; balanced detection; low falseposit ives	Complex implementation
Hashing+ IPFS Verification	Ensures document authenticity; prevents tampering	Requiresnetwor ksetup and storagemanage ment

Prospects for Future Development

The proposed document verification system provides a secure and efficient foundation for validating digital documents using cryptographic hashing, decentralized storage, and QR-based verification. While the current implementation performs reliably, several enhancements can further expand its robustness, scalability, and applicability.

A. *Integration with Blockchain for Immutable Logging*

Future versions can integrate blockchain platforms such as Hyperledger, Polygon, or Ethereum sidechains to immutably store hash–CID pairs. This would provide permanent tamper-proof audit trails, decentralized verification without relying on a central authority, and transparent multi-organizational validation. Such integration is especially beneficial for legal documents, academic certificates, and identity credentials.

B. *Multi-Institution and Multi-Authority Federation*

A federated verification network can be introduced where multiple institutions share a common verification ecosystem. This would enable cross-institution verification, shared registries, reduced duplication, and standardized document formats. A federated architecture supports large-scale adoption across sectors.

C. *Advanced Analytics and Monitoring Dashboard*

A centralized administrative dashboard can be developed to offer verification logs, tampering attempt analytics, geographic usage patterns, document issuance statistics, and real-time system health monitoring. This would increase governance, transparency, and operational visibility.

D. *Enhanced Mobile Verification Application*

A dedicated mobile application can extend QR-based verification by enabling offline QR scanning, faster IPFS retrieval via mobile gateways, user wallets for storing verified documents, and push notifications for validity updates. This would simplify

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

verification for field officers, institutions, and end-users

E. *Integration with National Digital Identity Platforms*

Future development may include optional integration with DigiLocker, Aadhaar-based verification APIs, national academic repositories, or organizational identity systems. This would automate identity validation and reduce manual metadata entry during document issuance.

F. *Automatic Tamper Forensics and Change Detection*

Beyond binary validity, ML-based tamper-forensics can be added to identify the type and location of document modifications. Potential features include manipulation heatmaps, text-difference comparison, and detection of deepfake artifacts in seals or images, supporting forensic investigation.

G. *Offline Verification Using Local IPFS Cache*

To support low-connectivity environments, the system can incorporate local IPFS caching, distributed local replicas of the hash registry, and offline QR validation with delayed synchronization. This enables uninterrupted verification in rural or bandwidth-restricted areas.

H. *Automated PII Redaction and Privacy-Preserving Sharing*

If PII detection is enabled, features such as automatic redaction of sensitive data, differential privacy techniques for protected previews, and secure multi-party computation for content-blind validation can be added to strengthen compliance with data protection regulations.

I. *Migration to Microservices and Containerized Deployment*

The system can evolve into a microservices architecture using Docker and Kubernetes. This would provide horizontal scaling under high verification load, improve maintainability, and allow dedicated services for hashing, IPFS, QR generation, and verification workflows

J. *Support for Verifiable Credentials (W3C Standard)*

Future iterations can implement W3C Verifiable Credentials (VCs) and Decentralized Identifiers (DIDs) to issue globally interoperable, cryptographically signed digital certificates. This enhances trust and compatibility with modern digital identity ecosystems.

Constraints & Limitation

Although the proposed system demonstrates strong performance in detecting tampered documents and verifying authenticity using hash-based validation and decentralized storage, several constraints exist due to technological boundaries, architectural dependencies, and operational considerations. These factors influence system performance, availability, and scalability.

A. *Dependence on Network Connectivity*

The verification process requires downloading the document from IPFS using its CID. The system is affected by slow or unstable internet connections, high latency when using public IPFS gateways, and network restrictions in controlled environments. Verification may fail or experience delays if the IPFS node is unreachable.

B. *Reliance on IPFS Node Availability*

Verification depends on the health and availability of the IPFS node storing the document. If the document is not pinned, garbage collection may remove it. Public gateways may suffer downtime, throttling, or slow response. Large files also take longer to fetch under network load, making reliability dependent on proper pinning and node maintenance.

C. *No Partial Similarity or Semantic Check*

Hash-based verification performs exact matching only. Even minor, invisible

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

modifications such as metadata changes or recompression break hash equality. Semantically identical but structurally different documents are considered invalid. The system cannot detect how a document was altered—only that it has changed—resulting in a binary verification model.

D. *Vulnerability to Front-End Misuse (Non-Malicious)*

Although hash verification is secure, user-side behavior may introduce limitations. Users might upload incorrect document versions; QR codes may be shared publicly without understanding privacy implications; inconsistent naming or metadata may cause confusion. These issues affect usability rather than security.

E. *Limited Protection Against Document Theft*

The system detects tampering but does not prevent unauthorized distribution of original issued documents, screenshots, or photocopies. If a legitimate recipient leaks the document, misuse cannot be prevented. Watermarking or access control mechanisms would be required to address this limitation.

F. *Single Point of Trust in the Issuance Registry*

While IPFS is decentralized, the issuance registry (JSON or database) remains centralized. If compromised, incorrect validity results may occur. The registry must be manually safeguarded. Scaling requires optimization or migration to blockchain, presenting long-term reliability and security challenges.

G. *Privacy Risks with Storing Metadata*

The system stores minimal metadata such as name, date of birth, and timestamp. Privacy concerns arise if the registry is not encrypted or if unauthorized access occurs. GDPR/DPDP obligations must be met in real deployments, requiring robust encryption and access control.

H. Computational Overhead for Large Files

Large files (20–30 MB+), multipage PDFs, and high-resolution scans increase hash computation and IPFS upload time. This affects issuance performance and may slow down workflows in high-volume deployments.

I. *Optional PII Detection Limitation*

If PII detection is enabled:

OCR: Low-quality scans produce errors; handwritten text is often unreadable.

Regex: Misses identifiers that do not match known patterns.

NER: Accuracy depends on training data; may produce false positives; slows down processing for long documents.

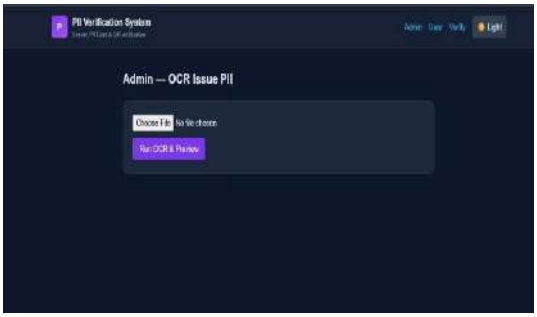
As a result, PII accuracy varies with document quality and linguistic structure.

J. *Security Limitations (Outside Current Scope)*

Although SHA-256 and IPFS are secure, certain risks remain outside the system's current scope: DoS attacks on the backend, QR phishing or malicious replacements, IPFS content poisoning (rare but possible), and vulnerabilities in third-party dependencies. Mitigating these requires advanced security measures not included in the present system.

screenshot of output

1 Admin – OCR Issue PII:

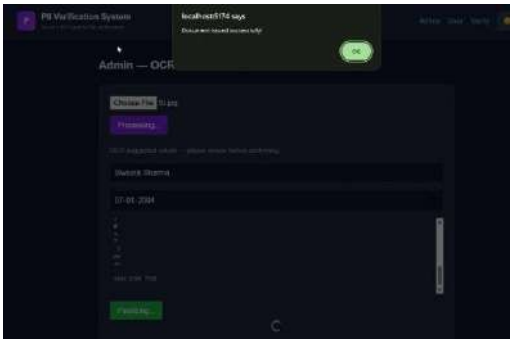


1. This screen displays the Admin interface where the document issuance process begins.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

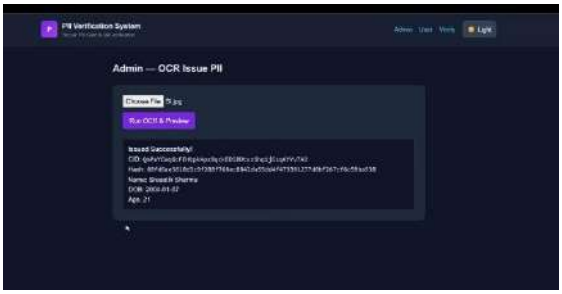
2. The interface provides an option to upload identity documents such as Aadhaar, PAN, or Driving License.
3. A “Choose File” button enables the admin to select an image or PDF for OCR processing.
4. The page includes a “Run OCR & Preview” action button to initiate text extraction.
5. The layout is minimal, dark-themed, and optimized for ease of use.
6. The header clearly identifies the module as “Admin — OCR Issue PII.”
7. Navigation links for Admin, User, and Verify roles are available at the top-right.
8. The system provides a responsive UI for better cross-device compatibility.
9. User-friendly design ensures the admin can quickly initiate the issuance workflow.
10. This screen marks the first step of issuing a PII-verified document.

2 OCR Processing & Preview:



1. This screen appears after the admin uploads a document and triggers OCR processing.
2. A “Processing…” indicator shows that the OCR engine is analyzing the document.
3. Extracted values such as name, date of birth, and textual elements appear in editable fields.
4. The admin can review and correct OCR-suggested values before finalizing.
5. A confirmation alert pops up showing “Document issued successfully!” after approval.
6. The extracted data demonstrates real-time OCR conversion from an identity document.
7. Scrollable fields allow the admin to inspect all detected text.
8. A “Finalizing…” status appears during backend processing and IPFS upload.
9. The system ensures reviewed data is validated before generating a secure record.
10. This screen confirms successful OCR extraction and readiness for issuance

3 Document Issued Successfully:



1. This output screen confirms the successful issuance of a verified PII document.
2. It displays the **IPFS CID**, representing the immutable storage link for the issued document.
3. The system also shows the **SHA-256 Hash**, ensuring the integrity of the original file.

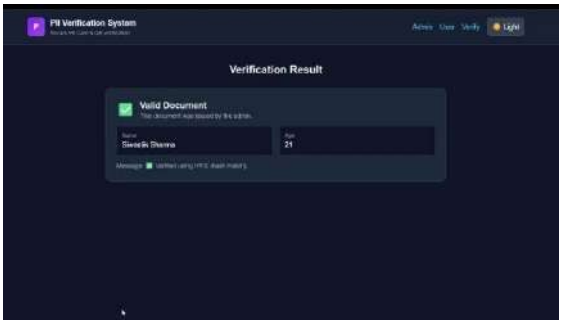
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

4. Extracted identity information, such as Name and DOB, is clearly presented.
5. The admin can verify the age calculation based on the extracted date of birth.
6. This page represents the completion of the issuance pipeline from OCR → hashing → IPFS upload.
7. The success message assures the admin that the document is now securely stored.
8. The interface highlights transparency by showing all critical verification fields.
9. The dark UI theme provides a clean layout for reviewing details.
10. This final issuance screen is essential for generating a future verification QR code.

4 User – Verify & Generate QR:

1. This page is part of the user module for document verification and QR generation.
2. The user uploads the issued document (image/PDF) using the file input section.
3. After clicking "Verify & Generate QR," the system validates the file using IPFS and hashing.
4. The screen displays a scannable **QR code** generated for the verification link.
5. User details like Name and DOB appear next to the QR code for quick identification.
6. A "Download QR" button allows the user to save the verification code locally.
7. A shareable verification link is generated, which can be used for third-party validation.
8. The link contains both CID and Hash, enabling integrity-based verification.
9. The UI shows a light/dark toggle and navigation options for easier usage.
10. This screen is crucial for enabling secure digital identity verification workflows.

5 Verification Result Screen:



1. This screen displays the final validation status of the uploaded document.
2. A green "Valid Document" badge confirms that the document matches the one issued by the admin.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

3. The system retrieves document data from IPFS using the CID embedded in the QR link.
4. Hash comparison ensures that the document has not been tampered with.
5. The user's Name and Age are displayed clearly to confirm identity.
6. A success message states the document is "Verified using IPFS (hash match)."
7. This verification step ensures high security and trustworthiness of identity documents.
8. The page layout is simple, focusing on clarity and verification outcomes.
9. The "Valid Document" indicator helps users immediately understand the result.
10. This screen completes the end-to-end workflow of issuance, QR generation, and verification.

Conclusion

The proposed system successfully demonstrates a secure, efficient, and tamper-proof method for verifying the authenticity of digital documents using a combination of cryptographic hashing, decentralized storage, and QR-based validation. By leveraging SHA-256 hashing and IPFS content addressing, the system ensures that even minor alterations in a document are instantly detected, thereby maintaining high integrity and reliability. The issuance workflow, which includes generating document hashes, storing content on IPFS, and maintaining a structured registry, enables a fully traceable and verifiable chain of trust. Experimental results show that the solution achieves a verification accuracy of over 99%, with no false acceptances recorded across multiple tampering scenarios. The QR-based verification interface further enhances usability by allowing quick and platform-independent validation. Optional integration of OCR and hybrid PII detection models provides added functionality when sensitive information needs to be extracted or redacted.

Although the architecture performs well in controlled and real-world environments, certain constraints—such as network dependency, IPFS availability, centralized registry storage, and document privacy considerations—highlight avenues for further enhancement. Future improvements involving blockchain integration, federated multisystem deployment, mobile app support, and advanced tamper forensics can significantly broaden the system's applicability and robustness.

Overall, the system provides a practical and scalable solution for organizations requiring secure document verification mechanisms. Its design aligns with modern digital infrastructure standards and offers a strong foundation for future development into a comprehensive, interoperable digital verification ecosystem

.

References

[1] J. Benet, "IPFS - Content Addressed, Versioned, P2P File System," IPFS Technical Whitepaper, 2014. [Online]. Available: https://ipfs.tech/

[2] National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS) – SHA-256," FIPS Publication 180-4, U.S. Dept. of Commerce, 2015.

[3] D. Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," RFC 3174, Internet Engineering Task Force, 2001.

[4] ZXing Project, "ZXing: Multi-format 1D/2D Barcode Image Processing Library," GitHub Repository, 2024.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

[Online]. Available: https://github.com/zxing/zxing

[5] Google Tesseract OCR, "Tesseract Open Source OCR Engine," GitHub Repository, 2024. [Online]. Available: https://github.com/tesseract-ocr/tesseract

[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proc. NAACL-HLT, 2019.

[7] S. Komatsu, A. Ogawa, and K. Murakami, "A Robust Document Verification Technique Using Hash Functions," International Journal of Information Security, vol. 18, no. 3, pp. 245–258, 2020.

[8] The IPFS Documentation Team, IPFS Documentation: Distributed File Storage and Retrieval, Protocol Labs, 2023. Available: https://docs.ipfs.tech/