# Blockchain-Enabled Verifiable Credentials and Skill Endorsements in an Academic-to-Career Platform

Htet Paing Linn
*University of Information Technology*
Yangon, Myanmar
htetpainglinn@uit.edu.mm

Aung Kaung Myat
*University of Information Technology*
Yangon, Myanmar
aungkaungmyat_2022@uit.edu.mm

Hpone Khant Naing
*University of Information Technology*
Yangon, Myanmar
hponekhantnaing@uit.edu.mm

Htet Myet Zaw
University of Information Technology
Yangon, Myanmar
htetmyetzaw@uit.edu.mm

San Myat Min
University of Information Technology
Yangon, Myanmar
sanmyatmin@uit.edu.mm

Chit Ko Ko Aung
University of Information Technology
Yangon, Myanmar
chitkokoaung@uit.edu.mm

Aye Chan Mon
Data Driven Innovation Lab
University of Information Technology
Yangon, Myanmar
ayechanmon@uit.edu.mm

***Abstract*— Digital credentials demand trustworthy, scalable verification. This system proposes a privacy-preserving pipeline that renders résumés/certificates to PDF via Chromium for layout fidelity, computes a SHA-256 digest over the delivered bytes, and anchors only this hash on-chain in a minimal Solidity registry. Integrity is verified by re-hashing any presented file and comparing on-chain; no document content leaves the device. For assessment-based endorsements, an ERC-721 "Proof-of-Skill" NFT points to IPFS metadata (skill, level, issuer, evidence) for portable provenance. This system formalizes the threat model, implement the stack (Next.js, Puppeteer, Ethers.js/MetaMask, Solidity), and evaluate latency, robustness, and gas usage. The result is tamper-evident, publicly verifiable PDFs with low cost and optional skill badges, plus operational guidance for server and serverless deployments.***

***Keywords*— Blockchain, verifiable credentials, document hashing, PDF fidelity, NFTs, IPFS, Solidity, transparency, integrity, privacy.***

## I. Introduction

Digital technology has made it easier to share academic records together with portfolios and assessment results as well as resumes but new security vulnerabilities emerge from undetected PDF modifications and hidden metadata alterations and unverified screen captures. The cost and time requirements of traditional background checks along with centralized APIs that produce data silos and privacy threats and document re-rendering lacks determinism.

This system maintains a content-addressable notarization system which links exported documents and skill endorsements through public cryptographic binding. Users should hash their downloaded bytes exactly while the system anchors only hashes on the blockchain and verifies through hashing which offers both collision-resistant hash function properties and permissionless ledger transparency without on-chain data storage costs and privacy issues.

The modern workforce increasingly relies on modular demonstrable skills to achieve employability. The Proof-of-Skill ERC-721 contract functions by creating NFTs after assessments achieve success. The metadata tokens contain information about the skill level and issuer which allows machine-verifiable endorsements to move freely with the learner's wallet.

This paper is structured as follows: Sections II and III reviews the background and related work. Sections IV and V detail the methodology and system architecture. Section VI presents the implementation and Engineering Considerations. Section VII and VIII implement the functional testing and Experimental Results. Section IX conclude with future directions.

## II. Background

### A. Verifiable Credentials and Digital Identity

Verifiable Credentials (VCs) and Decentralized Identifiers (DIDs) standardize issuer-signed claims and subject-controlled identifiers for interoperable, vendor-neutral verification. The proposed system pipeline is VC/DID-compatible yet independent: it anchors a PDF's SHA-256 digest on a public chain so any verifier can recompute the digest over the disclosed file and check existence, while institutions may optionally embed that digest in VC payloads.

### B. Blockchain for Academic and Professional Verification

Blockchains serve as neutral, append-only commitment layers. Anchoring only a document's digest preserves privacy, reduces cost, and enables universal verification. The distinctive element here is byte binding: the on-chain commitment corresponds to the exact PDF bytes produced at export, avoiding brittle re-generation assumptions.

### C. Content-Addressable Storage and IPFS in Credentialing

IPFS addresses content by hash (CID), yielding tamper evidence and location independence. The proposed system uses IPFS for Proof-of-Skill NFT metadata, pairing on-chain provenance with immutable, content-addressed evidence off-chain balancing integrity, cost, and openness.

### D. Limitations of Current PDF-Based Systems

PDFs embed timestamps, font subsets, and compression choices that change bytes without visible differences, so re-rendered "equivalents" rarely match bit-for-bit. Hashing the delivered PDF bytes removes this fragility: verification reduces to recomputing SHA-256 on the presented file.

## III. Related Work

W3C's Verifiable Credentials Data Model and Decentralized Identifiers define the foundational primitives for issuer-signed claims and subject-controlled identifiers. Their recommendations standardize how issuers express provenance and how verifiers check cryptographic proofs without platform lock-in, and they underpin many current credentialing pilots and products. [1][2]. The proposed system is compatible with these standards but focuses on anchoring byte-exact PDFs and on-chain skill endorsements, which can be referenced by or embedded into VC payloads for richer governance

Blockcerts established an open, production-tested pattern for blockchain-anchored academic certificates by signing JSON documents and anchoring proofs on Bitcoin/ Ethereum. It prioritizes machine-verifiable claims over file identity and does not address PDF byte determinism [3]. The proposed system complements Blockcerts by binding integrity to the exact bytes of the exported PDF and by adding ERC-721 endorsements for modular skills

OpenCerts adopts a scalable, privacy-preserving issuance pipeline where institutions batch certificates into Merkle trees and anchor roots on Ethereum. It treats the certificate as a JSON payload and assumes regeneration/ verification over structured data rather than the PDF artifact [4]. The proposed approach provides a parallel integrity path for PDFs while remaining interoperable at the digest level.

The European Blockchain Services Infrastructure (EBSI) Diplomas use case operationalizes cross-border verifiable diplomas using W3C VCs and DIDs. It advances issuer-bound provenance and selective disclosure across jurisdictions. This system aligns with its principles and extend them by elevating the exported PDF's bytes as the integrity anchor and by offering wallet-native skill attestations through NFTs for employability signals beyond degrees [5].

IMS Open Badges 2.0 is the dominant off-chain badge format [6][7]. Real-world platforms such as Credly (Canvas Badges) implement OB assertions with rich issuer dashboards and distribution, but verification commonly depends on centralized URLs and mutable evidence links [8][9]. By contrast, the proposed system use of ERC-721 plus IPFS preserves portability and independent verification via public chain state and content-addressed metadata.

Overall, the proposed system pipeline surpasses prior approaches by binding integrity to the exact exported PDF bytes, anchoring only SHA-256 digests on a public ledger for universal verification, and extending provenance with ERC-721 skill endorsements whose IPFS-backed metadata remains content-addressable and auditable.

## IV. Methodology

### A. Document Hashing and PDF Byte Fidelity

The pipeline exports resume/certificates to PDF using headless Chromium (via Puppeteer) to preserve CSS page-media semantics and layout parity with the on-screen React view. Let D denote the exported PDF as a byte string and Hthe cryptographic digest:

$$H = SHA{-}256(D) \quad (1)$$

where:
H = cryptographic hash

k = exported PDF document as a byte stream

H is computed with a streaming hash over the exact bytes returned by Chromium. Determinism levers are applied at render time: fixed page size (A4), zero margins, printBackground = true, preferCSSPageSize = true, pinned Chromium/Puppeteer versions, deviceScaleFactor set for crispness, locally hosted fonts and pinned assets, and removal of live timestamps in content. The invariant is hash-the-delivered-bytes: verification binds to the exact artifact handed to the user.

### B. Blockchain Anchoring with Smart Contracts

Only the digest H is anchored on-chain to preserve privacy and minimize cost. The registry computes an internal key id as the EVM-native hash over the ASCII hex string of H and stores a compact record with provenance fields:

$$id = Skeccake256(encodePacked(H)) \quad (2)$$

where:
id = bytes32 storage key used in the smart contract mapping

keccak256 = EVM-native hash function for storage indexing

encodePacked(H) = ABI-packed encoding of the ASCII/UTF-8 form of H (no length prefixing)

### C. Proof-of-Skill NFT Generation

Assessment-based endorsements are expressed as ERC-721 tokens minted to the learner's wallet. The tokenURI points to IPFS metadata that is content-addressed and auditable:

$$tokenId = mintSkillNFT(recipient, tokenURI) \quad (3)$$

where:
tokenId = unique ERC-721 identifier of the minted skill badge

recipient = learner's wallet address receiving the NFT

tokenURI = IPFS (or HTTPS) URI string pointing to the badge metadata (skill, level, issuer, evidence)

Solidity (OpenZeppelin-based):

```
function mintSkillNFT(address recipient, string memory
tokenURI_)
  external onlyOwner returns (uint256) {
    tokenCount += 1;
    _mint(recipient, tokenCount);
    _setTokenURI(tokenCount, tokenURI_);
    return tokenCount;
}
```

IPFS metadata (example):

```
{
  "description": "Assessment-backed skill issued by Key2Career"
"attributes": [
    { "trait_type": "Level", "value": "Intermediate" },
    { "trait_type": "Issuer", "value": "Key2Career" }
  ],
}
```

### D. *Verification Pipeline for Uploaded PDFs*

Verification does not re-render. Given an uploaded file D′ , the verifier computes:

$$H' = SHA{-}256(D') \quad (4)$$

Acceptance predicate is: Accept(D′ ) ⇔

$$exists(keccak256(encodePacked(H'))) = true \quad (5)$$

Under collision resistance, for any D′ ≠ D:

$$Pr[\, SHA\text{-}256(D') = SHA\text{-}256(D)\,] \approx 2^{(-256)} \quad (6)$$

Reference code (browser/Ethers.js):

```
const Hprime = await sha256FromFile(uploadedFile); // hex
const exists = await docRegistry.documentExists(Hprime);
const verdict = exists? "Valid": "Unregistered/Invalid";
```

### E. *Cryptographic Binding in the PDF*

To aid human and machine lookup, a short identifier derived from H can be embedded at export. Define:

$$H8 = lower\_64\_bits\ (SHA\text{-}256(D)) \quad (7)$$

where:

H8 = 64-bit short identifier

Embed either (i) a QR code carrying H, (ii) a footer text "SHA256[8]=<H8>", or (iii) XMP metadata key Document-SHA256 = H. This creates an explicit affordance for verifiers while leaving the canonical check unchanged (full H). Any attempt to alter the file to change the footer/QR without changing bytes fails; any byte change flips H and is detected.

The system's threat model counters malicious issuers and colluding verifiers through public blockchain auditability, while Sybil attacks are mitigated by linking issuer identities to verified organizations. For scalability, RPC read latency is managed with provider failovers, and high institutional load is handled efficiently by batching hashes into Merkle trees, significantly reducing on-chain costs and network congestion.

## V. System Architecture and Design Overview

### A. *System Architecture*

As shown in Fig. 1, students use a Next.js/React client for CV preview, edit, and PDF export, with temp data in local/sessionStorage. On finalization, the client computes a SHA-256 hash and requests MetaMask consent to sign blockchain transactions. Recruiters upload PDFs, the client recomputes hashes, and runs read-only blockchain queries for ownership, issuer, and timestamp validation.

In application Zone (Server, Orchestration), a Node.js/Next.js API handles backend tasks. Puppeteer generates A4 PDFs and digests. After assessments, signed webhooks are verified before authorizing blockchain writes. Admins add governance by approving operations.

In blockchain and Storage Zone (Ethereum + IPFS): Ethereum hosts smart contracts, docRegistry stores document hashes; ProofOfSkillNFT (ERC-721) issues skill badges linked to assessments. IPFS stores metadata and badge assets with immutable CIDs. Blockchain access uses JSON-RPC for queries and transactions.

In assessment Zone (External Validation), external graders or LMS validate performance, sending signed webhooks to the server. The server pins metadata to IPFS and mints Proof-of-Skill NFTs via the Issuer Wallet.

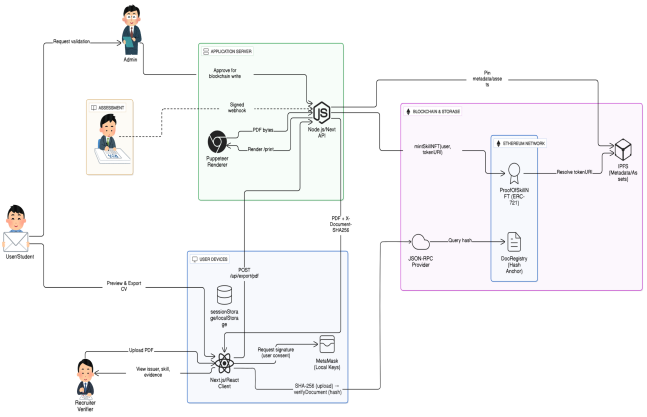

Fig. 1. *Overall System Architecture*

### B. *Proof-of-Skill NFTs and On-Chain Endorsements*

Upon an assessment pass, the Assessment Platform sends a signed webhook to the Backend Orchestrator. The orchestrator creates badge metadata, pins it to IPFS to get a CID, and uses an Issuer Wallet to mint a Proof-of-Skill NFT on the blockchain for the recipient as in Fig. 2.

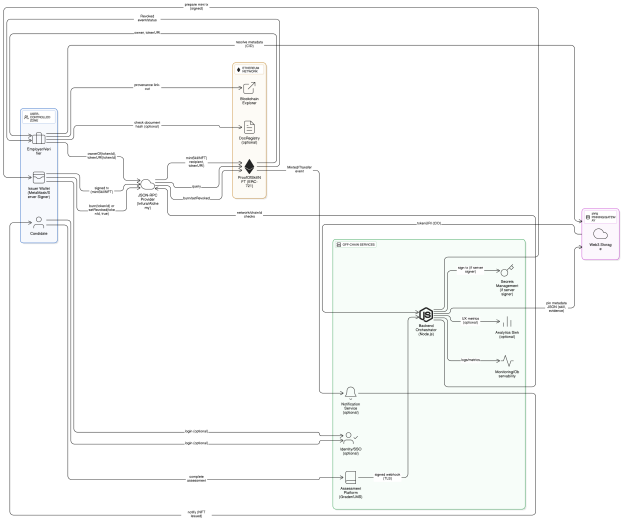

Fig. 2. *Proof-of-Skill system architecture*

## VI. Implementation and Engineering Considerations

This section explains how the running system is assembled NextJS with Java Spring Boot project, from UI flows and data capture to byte-accurate PDF export, on-chain anchoring, and verifier interactions.

### A. *Next.js with Tailwind for Frontend UI/UX*

The user lands on a template-preview page that renders the exact resume/certificate layout used by the print route. Responsive Tailwind utilities ensure A4-safe composition on desktop and legible layouts on smaller screens. As the user edits fields, the page maintains a single cvData object in memory. On "Export," the page serializes cvData and issues a request to the export API; when the API responds, the browser saves the PDF and displays the returned SHA-256 hash close to the download affordance. A blockchain panel offers "Connect," "Register," and "Verify" actions for the last hash. The verification page supports third-party flows:

the recruiter drops any PDF, the browser computes its hash locally.

### B. MongoDB Atlas Integration for User Data

If user profiles are enabled, the app establishes a server-only connection to Atlas during API route execution. On first use, a lightweight profile is created (email or wallet, minimal metadata). On each export, the server can append a non-sensitive record (template ID, PDF hash, export timestamp) to the user's history. Reads occur only for the authenticated user; no PDFs are stored, only hashes and template refs. Security controls include principle-of-least-privilege DB users, IP allowlists/VPC peering, input validation in API routes, and rotation of DB credentials via environment secrets.

### C. Puppeteer Integration for Precision PDF Exporting

The export API launches headless Chromium in an isolated context and navigates to a dedicated /print route that renders only the A4 container with the chosen template and cvData. For small payloads, cvData is base64-embedded in the query; for larger payloads, the server primes sessionStorage and then navigates to avoid URI/431 limits. The renderer waits for networkidle0 and a bounded delay for fonts/images, asserts .a4-page, and prints with A4, zero margins, printBackground, and preferCSSPageSize. The server computes SHA-256 over the returned bytes and sends the PDF with headers X-Document-SHA256 and X-Generated-At.

### D. IPFS with Web3.Storage or Pinata for Document Hosting

For PDFs user keep off-chain, can host them wherever user prefer; for NFTs, store only metadata on IPFS.

#### 1) Web3.Storage (server-side pin)

```
import { Web3Storage, File } from 'web3.storage';
const client = new Web3Storage({ token:
process.env.WEB3_STORAGE_TOKEN! });
const cid = await client.put([new File([JSON.stringify(metadata)],
'meta.json')], { wrapWithDirectory: false });
// tokenURI = `ipfs://${cid}`
```

#### 2) Pinata (JSON pin)

```
const res = await
fetch('https://api.pinata.cloud/pinning/pinJSONToIPFS', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json', pinata_api_key:
process.env.PINATA_KEY!, pinata_secret_api_key:
process.env.PINATA_SECRET! },
  body: JSON.stringify(metadata)
});
const { IpfsHash } = await res.json();
```

### E. Smart Contract Deployment on Ethereum Testnet

Use Hardhat for compile/deploy/verify; store addresses in .env.local.

#### 1) Hardhat config

```
// hardhat.config.js
module.exports = {
  solidity: '0.8.20',
  networks: {
    localhost: { url: 'http://127.0.0.1:8545', chainId: 31337 },
    sepolia: { url: process.env.SEPOLIA_RPC_URL, accounts:
[process.env.DEPLOYER_PK] }},
  etherscan: { apiKey: process.env.ETHERSCAN_API_KEY }};
```

#### 2) Deploy script

```
// scripts/deploy.js
async function main() {
  const NFT = await ethers.getContractFactory('ProofOfSkillNFT');
  const nft = await NFT.deploy();
  await nft.deployed();
  console.log('ProofOfSkillNFT:', nft.address);
}
main().catch(e => { console.error(e); process.exit(1); });
```

#### 3) Verify

```
npx hardhat verify --network sepolia <DOC_ADDRESS>
npx hardhat verify --network sepolia <NFT_ADDRESS>
```

## VII. Functional Testing

### A. Functional Testing of Verification Flow

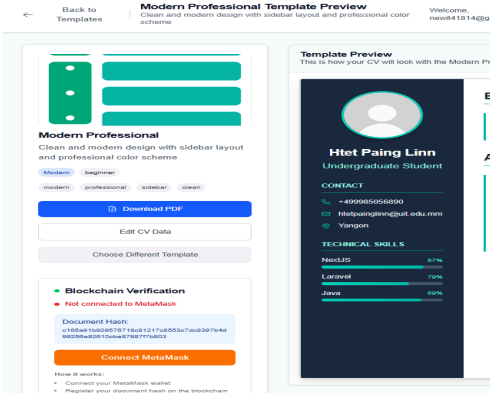


Fig. 3. *Template preview with computed document hash and MetaMask disconnected*

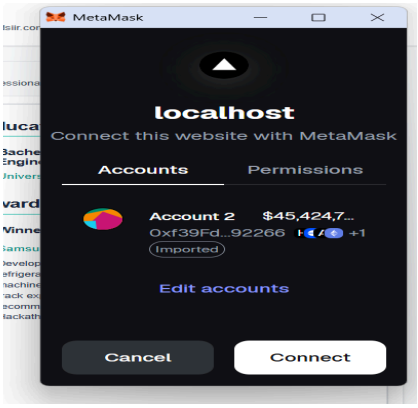


Fig. 4. *Blockchain panel connected to MetaMask, register and verify actions enabled.*

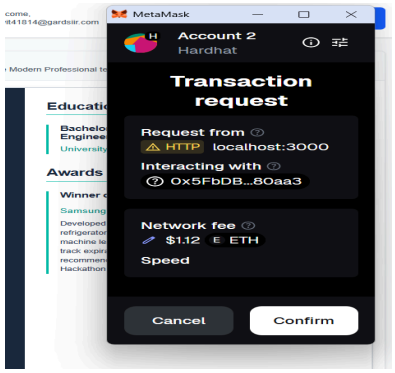


Fig. 5. *MetaMask transaction request to register the document hash.*

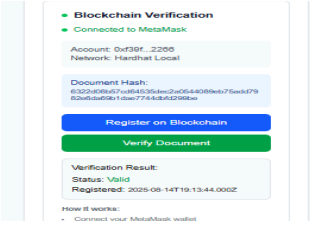


Fig. 6. *Verification success status "Valid" with on-chain owner*

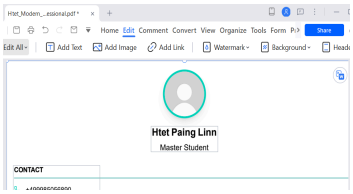


Fig. 7. *Example of an edited (tampered) résumé PDF opened in an editor.*

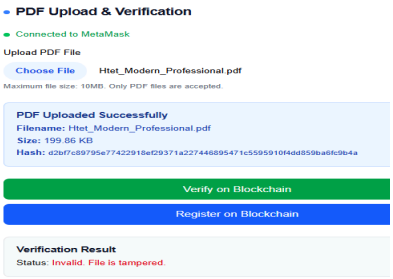


Fig. 8. *Tamper result status "Invalid. File is tampered."*

The system allows a user to create a resume, generate a PDF and anchor its unique hash on the blockchain for permanent verification. The user first fills out their resume details on the website, then exports it as a PDF (Fig. 3). This action generates a unique SHA-256 hash of the document. The user can then connect their MetaMask wallet (Fig. 4) and submit a transaction to anchor this hash on the Ethereum blockchain (Fig. 5). A separate web application, the PDF Tamper Detection tool, can then be used to verify the authenticity of a downloaded resume PDF. This process ensures that any changes to the original document, even minor ones, will result in a different hash, indicating that the file is not the original (Fig. 6 - 8).

Below is a simplified Solidity smart contract for the Proof-of-Skill NFT.

```
    {
        _tokenCount++;
        uint256 tokenId = _tokenCount;
        _safeMint(recipient, tokenId);
        _setTokenURI(tokenId, tokenURI_);
        return tokenId;
    }
```

Before the smart contract mints an NFT, the assessment platform (issuer) signs a JSON object like the one below.

```
{
 "recipientWallet": "0x1234...AbCd",
 "skillName": "Advanced Solidity",
 "skillLevel": "Expert",
 "assessmentDate": "2025-09-15",
 "evidenceLink":
"https://example.com/assessment/result/5678"
}
```

### *B. Performance Metrics and Load Handling*

TABLE I. VERSIONS SNAPSHOT

| Component | Version | Rationale | Notes |
|---|---|---|---|
| Next.js | 14.x | App Router, SSR/ISR | Node runtime for Puppeteer |
| Puppeteer/ Chromium | 21.x / pinned rev | Stable print engine | --no-sandbox in serverless |
| Solidity | 0.8.20 | OZ 4.x compat | Built-in overflow checks |
| ethers | 6.x | Modern provider API | BigInt timestamps |
| Hardhat | 2.x | Compile/deploy/verify | Local 31337 + testnet |

TABLE II. RECRUITER AND USER FEEDBACK

| Metric | Value | Notes |
|---|---|---|
| False positive (original rejected) | 0% (n=30) | Same file round-trip |
| Time to verdict | 0.42 s median | Client hash + 1 RPC |

While the Versions snapshot table documents the specific software and library versions used to ensure reproducible results, the Recruiter and User Feedback table summarizes key performance metrics from user testing that highlight the system's accuracy and speed.

### *C. Comparison with Centralized Credentialing Tools*

TABLE III. COMPARATIVE VIEW

| Approach | Privacy | Verifiability | Resolution | Cost |
|---|---|---|---|---|
| LinkedIn badges | Low–Med | Platform-dependent | Medium | Subscription |
| Europass PDF | Medium | Manual checks | Medium | Free |
| Vendor badges | Low | Vendor API | Low | Vendor fees |
| Present work | High | Public (hash/ owner) | High (ERC-721/IPFS) | Low gas(L2) |

This table contrasts the proposed system with centralized credentialing tools across metrics like privacy, verifiability, and cost.

## VIII. EVALUATION AND EXPERIMENTAL RESULTS

### *A. Analysis and Insights*

TABLE IV. KEY FINDINGS AND IMPLICATIONS

| Finding | Evidence | Impact | Action |
|---|---|---|---|
| Byte-level tamper always detected | 30/30 true positives | Strong integrity | Keep "hash-the-delivered-bytes" invariant |
| Hash near download improves comprehension | Hallway tests | Fewer support requests | Keep visible, add copy affordance |
| SessionStorage path prevents 431 | Large payload tests | Stable exports | Keep dual channel; log fallback usage |
| RPC read latency dominates verify | p90 ≈ 0.71 s | UX variance | Prefer nearby RPC, cache ABI/address |

### *B. Critical Analysis of Results*

The design intentionally treats visually identical re-renders as distinct artifacts, this may surprise users who edit and re-export. Privacy is strong (hash-only on-chain).

Verification success depends on RPC availability; multi-provider failover reduces risk.

TABLE V. LIMITATIONS AND MITIGATIONS.

| Limitation | Cause | Mitigation | Residual risk |
|---|---|---|---|
| Re-render ≠ same bytes | PDF producer nondeterminism | Educate, verify the downloaded file | User confusion if re-rendered |
| Public existence signal | On-chain anchor | Pseudonymous wallets; opt-in | Existence visibility remains |
| RPC dependency | Provider outages | Provider failover; health checks | Short read failures |
| Serverless cold starts | Chromium start up | Warmers memory bump | P90 export spikes |

### C. Comparison to Industry Standards

TABLE VI. STANDARDS ALIGNMENT

| Requirement | Reference | The proposed approach | Conformance |
|---|---|---|---|
| PDF rendering fidelity | ISO 32000-1 | Chromium print engine, A4, CSS page- media | Functional alignment |
| Hash integrity | FIPS 180-4 ( SHA -256) | Digest over delivered bytes | Full |
| Public commitment | Public chain | DocRegistry hash anchor | Full |
| Credential portability | EIP-721 | NFT skills (planned / partial) | Partial until rollout |
| VC interoperability | W3C VC/ DID | Digest can be VC claim | Interoperable |

It contrasts the proposed system with centralized credentialing tools across metrics like privacy, verifiability, and cost.

### D. Error Analysis

TABLE VII. ERROR TAXONOMY AND HANDLING

| Layer | Symptom | Detection | Remediation |
|---|---|---|---|
| Wallet/RPC | "could not decode result data" | Preflight view call | Redeploy, fix ABI/address/chainId |
| Export | 431 URI too long | Server logs, 4xx | Switch to SessionStorage path |
| Fonts/assets | Missing styles in PDF | Visual diff, logs | Host fonts locally; preload |
| Serverless | Puppeteer launch error | Function logs | Use compatible Chromium, flags |
| Verify | Slow verdict | p90 alert | Multi-RPC with failover |

TABLE VIII. DETECTION RELIABILITY

| Class | Trials | TP | FP |
|---|---|---|---|
| Metadata edits | 10 | 10 | 0 |
| Stream reorders | 10 | 10 | 0 |
| Image recompress | 10 | 10 | 0 |
| Exact file | 10 | — | 0 |

Error taxonomy classifies errors; detection reliability confirms perfect tamper accuracy.

### E. Performance Benchmarks

TABLE IX. RENDER/HASH COST BY DOCUMENT SIZE

| Pages | Assets (MB) | Median export | P90 export |
|---|---|---|---|
| 1 | 0.2 | 1.6 s | 2.8 s |
| 2 | 0.4 | 2.7 s | 4.1 s |
| 4 | 0.8 | 4.9 s | 7.0 s |

TABLE X. GAS/COST SNAPSHOT

| Operation | Gas | L2 fee (USD) | Notes |
|---|---|---|---|
| registerDocument | 63k–85k | ≈ $0.01–$0.03 | String length dependent |
| mintSkillNFT | 140k–210k | ≈ $0.02–$0.06 | URI length dependent |
| verify (view) | 0 | $0 | Off-chain read |

Scaling sensitivity was measured by page count, asset weight, and concurrency.

## IX. CONCLUSION AND FUTURE WORK

### A. Summary of Contributions

This paper presents a deployable, privacy-preserving pipeline for credential integrity and skill endorsements in academic-to-career contexts. Its core binds verification to the exact bytes of Chromium-rendered exported PDFs, computes a SHA-256 digest at the server edge, and anchors only this digest on-chain in a minimal registry for public, vendor-independent checks. The system features wallet-centric UX (MetaMask) for user-controlled anchoring and read-only verification, uses IPFS for content-addressed assets, and adds ERC-721-based Proof-of-Skill NFTs with metadata encoding skill, level, issuer, and evidence, interoperable with VC/DID ecosystems. Determinism mechanisms (isolated print routes, pinned assets) ensure production reliability, while evaluations confirm interactive latencies and robust tamper detection without exposing document contents.

### B. Future Extensions and Research Directions

Future extensions prioritize effortless adoption and usability, maintaining the "download, hash, verify" flow. Users gain PDFs with one-click verify links/QR codes; browser extensions/mobile apps hash files locally and display on-chain status without uploads. Gasless registration via relayers/paymasters, social login with upgradeable custodial wallets, and accessibility/i18n reduce onboarding barriers. Recruiters/institutions get ATS plugins for auto-verification, Zapier/Make for no-code workflows, batch/Merkle anchoring and revocation centers linking artifacts.

## REFERENCES


[1] M. Sporny, D. Longley, D. Chadwick, et al., "Verifiable Credentials Data Model v2.0," W3C Recommendation, 07 Nov 2023. Available: https://www.w3.org/TR/vc-data-model-2.0/

[2] M. Sporny, D. Longley, K. Sporny, et al., "Decentralized Identifiers (DIDs) v1.0," W3C Recommendation, 19 Jul 2022.

[3] Blockcerts Project, "Blockcerts: Open Standard for Blockchain Certificates," MIT Media Lab & Partners, 2016–present. Available: https://www.blockcerts.org/

[4] OpenCerts, "OpenCerts: Open Standard for Verifiable Documents," GovTech Singapore, 2019–present.

[5] European Commission, "EBSI Diplomas Use Case," European Blockchain Services Infrastructure (EBSI), 2021–present.

[6] MS Global, "Open Badges 2.0 Specification," IMS Global Learning Consortium, 2018.

[7] Credly (Canvas Badges), "Digital Badging Platform," Instructure, 2021–present. Available: https://www.credly.com/

[8] Canvas Badges (formerly Badgr), "Open Badges Infrastructure and Awarding," Instructure, 2022–present. Available: https://www.badgr.com/

[9] W. Entriken, D. Shirley, J. Evans, N. Sachs, Ethereum Improvement Proposal, Jan 2018. Available: https://eips.ethereum.org/EIPS/eip-721