Blockchain-Based Smart CV for Verified Work Credentialing

Context

This project addresses a key challenge in the human resources (HR) and recruitment sector, focusing initially on the European labor market. It particularly targets industries where professional credibility is critical—such as technology, healthcare, marketing, and finance—where roles often require verifiable, specialized skills.

According to Eurostat (2025), 62% of companies in the European Union have integrated digital recruitment tools into their hiring processes. However, the sector continues to suffer from inefficiencies in verifying candidate credentials. GDPR regulations further complicate cross-border data handling, reinforcing the need for secure and privacy-preserving solutions. This project targets two core user groups:

- **Job seekers**: Individuals who need a standardized and trustworthy way to prove their professional background.
- **Employers**: Companies that require efficient and secure systems to validate candidate credentials quickly and reliably.

Despite increasing digitization, the market lacks a unified, tamper-proof standard for CV validation. This opens an opportunity for blockchain to introduce a decentralized and verifiable system for certifying work and educational history.

Problem Definition

Today, verifying a candidate's resume is a **time-consuming, costly, and error-prone process**. Existing background checks often rely on manual methods or centralized platforms, which are vulnerable to fraud and inefficiency.

- **Resume fraud is widespread**: Studies estimate that around **56%** of workers and nearly **47% of Gen Z applicants** admit to misrepresenting their experience on resumes or job applications[1].
- **Verification is slow**: Employment verification typically takes **1–5 business days**[2].

[1] HRO Today. https://www.hrotoday.com/news/over-half-of-employees-report-lying-on-resumes/; NY Post. https://nypost.com/2025/05/18/lifestyle/gen-z-population-have-lied-on-job-applications-half

[2] Clarifacts. https://clarifacts.com/faqs/employment-verification-turnaround-time/; Truework. https://www.truework.com/verifications/knowledge/employment-verification/verification-101-how-long-does-it-take

- **Costs are high**: Background checks cost between $29 and $75 per candidate[3].
- **Job scams are rising**: In 2024 alone, job seekers lost over $220 million to fraudulent job postings and gamified scams[4].
- **No global standard**: Credential verification lacks a universally accepted standard, leading to duplicated efforts and low trust between recruiters and applicants.

These issues create bottlenecks in hiring, reduce labor market efficiency, and increase the risk of poor hires.

Proposed Solution: Blockchain-Enabled Smart CV

We propose a decentralized application (DApp) that enables **verifiable, immutable, and secure certification of CV content**—creating what we call a **Smart CV**.

Terminology

- **Reliable proof / certificate**: A digitally signed attestation of an experience or credential.
- **Experience**: Any professional or educational activity (e.g., jobs, internships, seminars).
- **Standard**: Formal qualifications (e.g., degrees, diplomas, certificates).
- **Certifier agent**: An authorized entity (e.g., company, university, training center) that issues certificates.

Solution Overview

Our system will allow users to:

- **Create/Store** verifiable proof of experiences and standards.
- **Verify** any certificate using a blockchain lookup mechanism in reasonable time.
- **Link** certifications to a unique Smart CV, usable across applications and platforms.

The Smart CV ensures that recruiters can instantly verify key credentials without relying on centralized databases or manual checks.

As defined in Guesmi et al. (2023)[5], "Smart CV" refers to a digital resume containing cryptographically certified claims about the applicant's background.

Revenue Model

The project introduces a **token-based revenue model**, with two main income streams:

- **Subscription fees**: Paid by companies or institutions for access to the verification service.

[3] iProspectCheck. https://iprospectcheck.com/best-background-checks/

[4] FTC. https://www.ftc.gov/news-events/data-visualizations/data-spotlight/2024/12/paying-get-paid-gamified-job-scams-drive-record-losses

[5] *Smart CV for Lifelong Qualifications Certification Based on Blockchain*, https://www.researchgate.net/publication/367893845_SMART_CV_FOR_LIFELONG_QUALIFICATIONS_CERTIFICATION_BASED_ON_BLOCKCHAIN

- **Pay-per-verification**: Individual verification requests processed through tokens.

The token-based design supports:

- **Token-supported revenue model**: Tokens are used to mediate access and rewards, creating a flexible and scalable payment layer on top of a subscription-based service.
- **Incentive alignment**: Certifier agents earn tokens for issuing certificates, reducing their subscription fees and encouraging participation.

Technical Implementation

The system architecture integrates blockchain technology as a secure and auditable control layer, while encrypted user data is stored off-chain in a managed database. This hybrid approach guarantees performance, data confidentiality, and immutability of key records.

Key Components

- **Encrypted Off-chain Database**: All user information (CV data, certifications, experience) is stored off-chain in a secure, encrypted database managed by the platform. Only metadata and verification hashes are stored on-chain.
- **Blockchain Layer**: The blockchain is used to store cryptographic hashes of records and control data access via smart contracts. This ensures data integrity, non-repudiation, and traceable access control.
- **Smart Contracts Suite**: Several smart contract serivces are deployed to manage the certification lifecycle and token-based economy:
  - `CertificateRegistry`: Stores the hash of each certificate or experience and links it to a Smart CV ID.
  - `DeployCertificate`: Enable the process of Certificate creation and interact with the storage of the latter.
  - `AccessManager`: Controls who can submit, update, or revoke certifications using a permission model and enforces compliance rules (e.g., GDPR).
  - `TokenManager`: Implements a utility token used as currency within the system.
  - `IncentiveManager`: Distributes tokens to certifiers who upload verified records and to users who contribute valid, verifiable information.
  - `SearchAccess`: Allows recruiters to query candidate profiles by spending tokens and retrieving encrypted record references via hash lookups.
- **DApp Frontend**: A decentralized application (DApp) provides the user interface for uploading data, viewing certifications, requesting access, and performing job-related searches.
- **Certifier Onboarding and KYC**: Entities that issue certifications (e.g., universities, companies) must complete a Know Your Customer (KYC) process to be whitelisted as trusted certifiers on-chain.

Privacy and Compliance

By design, no personal data is stored directly on-chain. Hashes are non-reversible and serve as proof of existence and integrity only. Full compliance with GDPR is achieved by allowing users to control and revoke access to their data, while ensuring the traceability and transparency of modifications through blockchain anchors.

Certificate Creation Process Summary

The following outlines the process for creating a certified, verifiable credential using off-chain user interaction, IPFS storage, and on-chain verification via smart contracts.

Step 1: Claim Creation (Off-Chain)
A user (candidate) creates a claim with the following structure:

- `metadata`: Description of experience or standard.
- `certifier_id`: Address of the certifier entity.
- `user_id`: Wallet address or decentralized ID of the user.
- `user_signature`: User's digital signature over the claim.

Step 2: Certifier Verification (Off-Chain)
The certifier reviews the claim. If approved, they sign it using their private key to authenticate the validation.

Step 3: Data Storage on IPFS (Off-Chain)
The signed claim is encrypted or masked and uploaded to IPFS. The system obtains a content identifier (CID).

Step 4: Hash Generation (Off-Chain)
A cryptographic hash (e.g., SHA-256) of the encrypted claim.

Step 5: On-Chain Certification (Blockchain)
The smart contract stores in the blockchain:

- `certificate_hash`
- `IPFS CID`
- `timestamp`

Step 6: Incentive Mechanism (Blockchain)
The system rewards the certifier agent with platform tokens as an incentive for participating in the certification process.

This process ensures that each credential is immutable, verifiable, and privacy-preserving.

Certificate Verification Process Summary

The verification process allows an authorized entity (e.g., employer or recruiter) to confirm the validity and authenticity of a certificate linked to a Smart CV.

Step 1: Verification Request (Off-Chain)
An entity initiates a verification request by submitting:

- Proof of identity (identificator of the entity to check its permission)
- The identifier of the certificate to be verified

Step 2: Subscription Check (On-Chain)
The smart contract verifies that the entity has an active subscription or valid access rights.

Step 3: Token Payment (On-Chain)
The verifier transfers tokens to the system in exchange for the certificate verification service. This is handled via a smart contract function.

Step 4: Certificate Lookup (On-Chain)
The smart contract retrieves certificate data using the submitted identifier.

Step 5: IPFS Retrieval and Decryption (Off-Chain)
The system fetches the encrypted certificate data from IPFS and it decrypts the content.

Step 6: Display to Verifier (Off-Chain)
The verified data is presented to the empolyer to allow quick verification.

This process ensures verifiability, audability, transparency, and privacy-preserving access to trusted CV credentials.

Minimum Viable Product

Some consideration: proposition of the actual implementation.

We will develop the core mechanism of this system as a framework of functionalities alongside a simple web application interface to test this capabilities, in particular we will focus on the on-chain smart contract services that we discussed as foundational.

Services (or functionalities) we will develop:

- **Store Certificate** store the information about certificate to store on the chain, and build a mock function that should store the information on the IPFS infrastructure. This includes managing the incentive mechanism, i.e. tokens exchange to the certifier
- **Verify Certificate** manage the retrieval of the proof and the retrieval of the actual information. This includes also tokens exchange and subscription verifier.

For the purpose of the project actual implementation we will use a toy static list of whitelisted entity certificator, and we will create some toy user to show the system core's functionality.

The future improvement would be to implement also a minimum UX framework and to expand the functionalities as for the actual creation of the proof and the managing of the IPFS's storaging services.

Technical Implementation Summary (Structured)

The SmartCVs MVP DApp consists of two main interactive processes: certificate creation and certificate verification. These processes involve multiple off-chain mock components and two real on-chain smart contract functions. The architecture is divided into the following core modules:

- **Smart Contracts:**
  - `storeCertificate(string cid, bytes32 hash, address user)` — Stores the certificate hash and IPFS CID on the blockchain.
  - `verifyCertificate(bytes32 certId)` — Retrieves certificate data by ID and checks access rights.
  - Internal mappings to track:
    * Whitelisted certifiers and verifiers
    * Mock token balances for incentives and access control
- **Front-End Web Application (React + ethers.js):**
  - Handles user interactions (form inputs, UI display, MetaMask connection)
  - Simulates off-chain actions like signing, encryption, and IPFS storage
  - Displays stored and verified certificate information

Component-wise Technical Responsibilities:

Component	Functionality	Implementation Notes
Candidate (User1)	Creates a certificate claim	• Inputs metadata, certifier ID, wallet address • "Signs" the claim using a client-side hash (e.g., SHA-256 of JSON string)
Certifier (User2)	Approves the certificate	• Reviews mock claim • Simulates approval by adding a hash signature field
IPFS Storage (Mock)	Stores encrypted certificate data	• CID is mocked with a fixed string or dummy response • Optionally, upload real file to `web3.storage` and return real CID
Certificate Hash	Ensures data integrity	• Computed using SHA-256 in browser • Sent to smart contract with the CID

Component	Functionality	Implementation Notes
Smart Contract: `storeCertificate`	On-chain certificate registration	• Verifies sender is a whitelisted certifier • Saves cert hash, CID, user ID, and timestamp • Increments token balance (mock reward)
Verifier (Employer)	Initiates verification	• Inputs certificate ID or metadata • Front-end calculates certificate ID from user address + CID
Smart Contract: `verifyCertificate`	On-chain certificate retrieval	• Confirms verifier is authorized • Deducts tokens from balance (mock) • Returns cert metadata, CID, issuer, timestamp
Frontend Certificate Viewer	Displays verified data	• Simulates IPFS fetch and "decryption" • Displays hash, CID, certifier, user, and timestamp

Key Technologies Used:

- `Solidity` for smart contract logic, with `RemixIDE`
- `Geth` for local blockchain deployment
- `React.js` for the web application frontend
- `ethers.js` for blockchain interaction via MetaMask
- `crypto-js` or `window.crypto.subtle` for hashing (if implemented
- `web3.storage` or mocked CID for IPFS handling

This architecture separates mock logic from real smart contract logic and allows future replacement of simulated parts (e.g., real encryption or IPFS) with production-ready tools.