Blockchain-Based System for Secure Talent Credential Verification
Final Report

Benjamin Vauchel *(40280381)*
Concordia University

Abstract

Verifying academic and professional credentials remains a critical task due to inefficiencies, forgery risks, and the absence of secure, standardized mechanisms for verification and sharing. This project presents a blockchain-based system for secure talent credential verification using Hyperledger Fabric. The solution leverages a permissioned network composed of three organizations—talents, institutions, and companies—to ensure decentralized governance, data integrity, and controlled access. Smart contracts were developed to manage credential creation, approval, querying, and revocation. A REST API and web interface were implemented to provide seamless interaction with the blockchain, abstracting its complexity from end-users. The system was evaluated on a local Fabric deployment, analyzing throughput and latency under different workloads and transaction sizes. Results suggest that the number of concurrent transactions has a significant impact on performance, whereas transaction size has negligible effect. This work demonstrates the potential of permissioned blockchains for verifiable credential management, while also highlighting the practical challenges related to deployment, scalability, and measurement accuracy. The source code for this project can be found at https://github.com/benjaminvauchel/hyperledger-fabric.

1 Problem Statement

Verifying academic credentials, certifications, and professional skills remains a significant challenge due to risks of forgery, inefficiencies in traditional processes, and the lack of secure, standardized methods for sharing and verifying these credentials. Traditional paper-based systems are slow, prone to human error, and susceptible to falsification. Although modern computer-based systems offer improvements in speed and accessibility, they often rely on centralized databases, which are vulnerable to data breaches, unauthorized modifications, and a lack of transparency across stakeholders.

1.1 Motivation and Relevance

Reliable and tamper-proof talent verification is essential for employers, educational institutions, and credential holders. A secure and efficient verification mechanism can streamline recruitment, reduce administrative overhead, and foster trust between parties. By addressing the limitations of existing systems, a decentralized solution can improve hiring practices, lower recruitment costs, and enhance the credibility of individual qualifications in a scalable and transparent manner.

1.2 Proposed Solution

A blockchain-based solution offers a decentralized, secure, and tamper-resistant approach to credential verification. The proposed system leverages Hyperledger Fabric to create a permissioned blockchain network where talents, institutions, and companies interact under well-defined roles. Smart contracts are employed to manage the credential lifecycle, including submission, approval, revocation, and querying of credentials. Privacy, scalability, and usability are core concerns in the system's design. The architecture ensures that talents retain control over their own data, while institutions and companies are able to verify credentials without exposing sensitive information.

1.3 Related Work

Several blockchain-based systems have been proposed to address credential verification challenges. Khan and Ahmad (2022) designed a blockchain system using smart contracts and IPFS for academic degree verification, although it remained unimplemented, leaving questions about its practical viability [3]. Blockcerts [5] and OpenCerts [4] are notable projects that use public blockchains (Bitcoin and Ethereum, respectively) to issue and verify certificates, providing open standards and libraries for adoption. Bhumichitr and Channarukul (2020) proposed AcaChain, a Hyperledger Fabric-based academic credential system using hashed references

and mobile-based access control [1]. Tariq et al. (2019) developed Cerberus, focusing on scalability, selective data disclosure, and credential revocation through smart contracts and multi-signature mechanisms [6]. More recently, Ileana and Popgeorgiev (2024) analyzed the role of blockchain in enhancing transparency and trust in education by enabling secure diploma issuance and fraud prevention [2]. These works inform the design of this project and highlight the importance of balancing security, privacy, and usability.

1.4 Implementation Overview

The system is built on a local deployment of Hyperledger Fabric using Docker. The network includes three organizations: Talents, Institutions, and Companies—each with a defined role. Talents initiate credential requests, which are then reviewed and approved by institutions. Once verified, credentials are stored immutably on the blockchain. Smart contracts (chaincode) written in Go enforce the credential lifecycle rules and manage operations such as creation, approval, revocation, and querying. A REST API serves as the bridge between the blockchain network and the frontend interface, enabling users to interact with the system through standard web protocols. The frontend application allows talents to submit credentials and view their status, while institutions can approve or revoke them. The implementation phase is followed by testing and performance evaluation to ensure system scalability, reliability, and security.

1.5 Challenges and Limitations

Several challenges were encountered during the development process. First, scalability remains a concern, especially in managing a high volume of credential transactions without affecting network performance. Response times may also be impacted by the execution time of smart contracts and consensus mechanisms. Careful testing is required to address these performance bottlenecks. Another difficulty lies in evaluating the system under realistic conditions, requiring appropriate workloads and representative usage scenarios. Furthermore, since Hyperledger Fabric does not natively support credential revocation, a custom revocation mechanism had to be implemented as part of the smart contract logic.

2 Solution

To address the problem of securely managing and verifying talent credentials, we developed a blockchain-based system using Hyperledger Fabric. This permissioned network enables decentralized, tamper-resistant storage and verification of academic and professional credentials while maintaining access control among stakeholders (talents, institutions, and companies).

2.1 Network Architecture

The architecture was designed to reflect real-world interactions between three types of organizations: talents (individuals claiming credentials), institutions (academic bodies issuing, verifying and consuming credentials), and companies (entities issuing, verifying and consuming credential data). The final network, shown in Figure 1, is structured to include three organizations, each hosting at least one peer node, and a RAFT-based ordering service with three orderer nodes to maintain consensus and fault tolerance. All participants interact over a shared channel, with plans to integrate Private Data Collections (PDCs) for more granular data privacy in future enhancements. All three organizations have a Certificate Authority that has generated the necessary certificates for the nodes, admins, organizations definitions, and applications of its organization. The channel configuration has been agreed by the organizations, and it records the organizations authorized to participate in the channel and outlines the policies that govern decision-making processes and determine how outcomes are achieved. We configured CAs for each organization to manage identity issuance and authentication. These CAs issued X.509 certificates used to identify components as members of their respective organizations and to enable secure transaction signing and endorsement. We also defined the corresponding Membership Service Providers (MSPs), which link issued certificates to organizational identities and play a key role in enforcing access control policies across the channel. This setup ensures that all entities in the network are properly authenticated and authorized to participate in blockchain operations. As indicated in Figure 1, the institution organization runs one peer and owns the ordering service. The talent organization joins two peers to the channel, and the company organization joins one peer to the channel. Peers host both the ledger and the chaincode, while the ordering service collects endorsed transactions from client applications, arranges them into blocks, and distributes these blocks to all peers in the channel. Each committing peer then records the transactions and updates its local copy of the ledger accordingly. Every node in the channel stores a copy of the ledger of the channel, L1, which will be updated with each new block. After the ordering service was joined to the channel, we deployed the chaincode C1.

2.2 Smart Contracts

In Hyperledger Fabric, the chaincode defines the business logic of the system and govern all credential-related operations. Two types of credentials are supported: academic and professional, each associated with a common set of attributes such as `CredentialID`, `FirstName`, `LastName`, `TalentID`, `Skills`, `VerificationStatus`, and `VerifiedBy`. A variety of chaincode functions were implemented to manage the full credential lifecycle, as shown in Table 1. Members of the

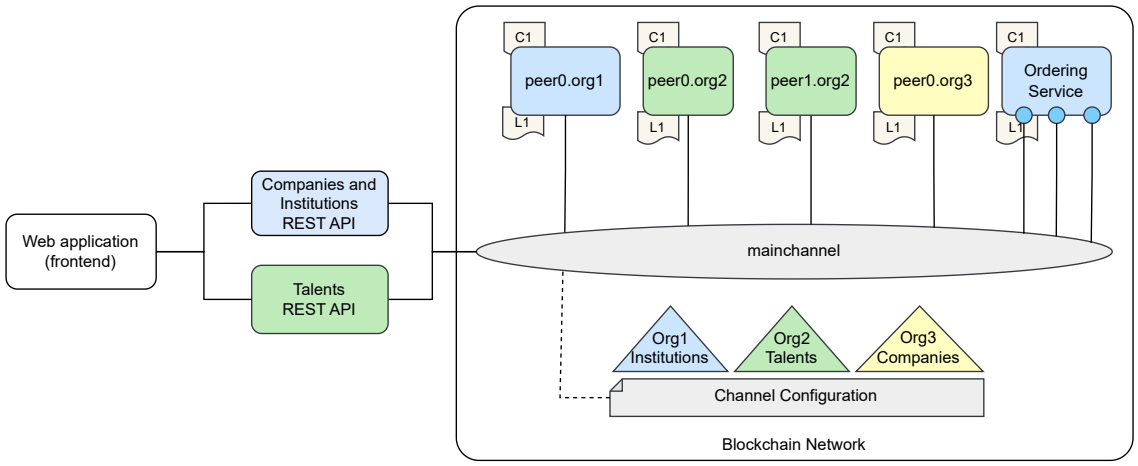

Figure 1: Network Architecture.

talent organization cannot update the verification status of a credential, meaning they cannot approve or revoke it, whereas companies and institutions can. The modularity of the smart contracts also leaves room for adding more asset types or expanding existing attributes.

Table 1: Main Smart Contracts for Credential Lifecycle

Operation	**Functions**
Initialization	`InitLedger`
Creation	`CreateAcademicCredential`, `CreateProfessionalCredential`
Existence	`CredentialExists`
Verification	`UpdateVerificationStatus`
Deletion	`DeleteTalentCredential`
Querying	`GetAllCredentials`, `GetBaseCredential`
Update	`UpdateSkills`, `UpdateName`

To enable secure and collaborative execution of smart contracts, the Fabric chaincode lifecycle introduces a structured and decentralized process through which organizations on a channel can agree upon and manage the deployment of chaincode. It begins with (1) packaging the chaincode, where the chaincode is bundled into a .tar.gz archive containing a metadata file and the source code. This package is identified by a label and can be created independently by each organization or shared to ensure consistency. Next, (2) installing the chaincode involves deploying the package on each peer that will execute or endorse transactions. This step generates a package identifier used in the next phase. In (3) approving the chaincode definition, each organization endorses a definition specifying the chaincode name, version, sequence number, endorsement policy, and optional settings such as private data collections. Once enough organizations have approved, the definition can be committed. In step (4), committing the chaincode definition, an authorized organization submits a transaction that finalizes the definition on the channel, triggering the launch of the chaincode container on the relevant peers. Finally, (5) initializing the chaincode may be required if the definition specifies an Init function, which is our case. In that case, the first invocation of the chaincode must call this function to set up any initial state, and the call must meet the endorsement policy. This lifecycle ensures that all channel members are aligned on how the chaincode operates before it is activated on the network.

In support of deployment and testing, a set of scripts was written to automate the chaincode lifecycle. These scripts handle packaging, installation, approval collection, and committing of chaincode definitions, as well as execution of test transactions via `peer chaincode invoke` and `query` commands. This automation accelerates development and reduces the risk of human error in manual configuration. For testing, the `peer` commands allow us to interact with a peer, and invoke the smart contracts.

2.3 REST API

To enable seamless integration between the blockchain network and the user-facing application, a REST API was developed using the Hyperledger Fabric Gateway SDK. The API performs authentication checks, validates payloads, and triggers smart contract functions. It offers many endpoints, as described in Table 2. Fields must be given in the request body as JSON for POST, PUT and DELETE methods, and in the URL as query parameters for the GET method.

The API includes field validation (required fields, correct formats) and role-based access checks (e.g., only institutions and companies can approve credentials).

The initial plan was to run three backend servers in parallel, one for each of the three organizations. In practice, one server

Table 2: REST API Endpoints for Talent Credential Verification

Method	Endpoint	Description
POST	/credentials/academic	Create a new academic credential
POST	/credentials/professional	Create a new professional credential
PUT	/credentials/{id}/approve	Approve a credential with the specified ID
PUT	/credentials/{id}/revoke	Revoke a credential with the specified ID
DELETE	/credentials/{id}	Delete a credential with the specified ID
PUT	/credentials/{id}/skills	Update the skills field of a credential
PUT	/credentials/{id}/name	Update the name (first and last) of a credential
GET	/credentials/{id}	Retrieve a credential by ID and type (supports query param `?type=academic|professional|base`)
GET	/credentials/all	Retrieve all credentials
GET	/credentials	Query credentials using standard query parameters
GET	/credentials/query	Perform a custom query with chaincode function and arguments

is dedicated to the talent organization, while a second server handles both the institution and company organizations.

To test the implementation of the backend application, we used cURL to send different requests.

2.4 User Interface

On the frontend side, a web interface was developed to allow users to interact with the blockchain-based credential system in an intuitive and user-friendly manner. The application enables talents to submit academic or professional credential requests through structured forms, while institutions can review, approve, or revoke credentials with a single click. The interface is connected to the backend REST API and dynamically displays credential information retrieved from the blockchain. Additional features include visual feedback during operations (e.g., loading states, error messages), user role-specific actions, and basic validation for input fields to prevent malformed submissions. This frontend is designed to abstract the complexity of the underlying blockchain while clearly presenting the available actions to all users. It is not yet ready for production deployment, but serves as a user-friendly application for demonstration purposes.

3 Results

The evaluation was conducted on a simplified blockchain network, as the deployment of the intended architecture resulted in an unresolved error. Consequently, the network used for testing includes three organizations, with two having a single peer each and one organization responsible for ordering, operating with a single orderer.

We first evaluated the throughput and latency of the Fabric network under different levels of transaction concurrency, using a total of 500 transactions. Specifically, we varied the number of parallel participants sending transactions, from

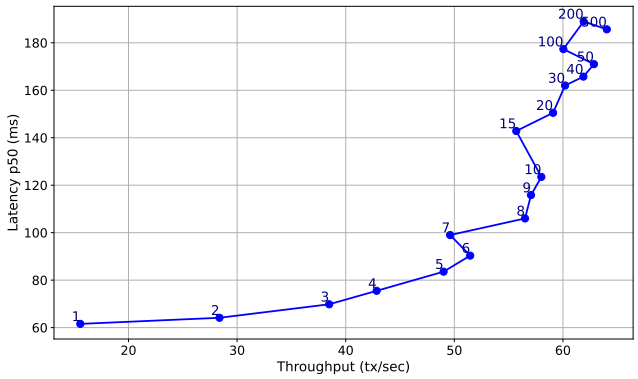

Figure 2: Throughput vs. Latency, with annotated values indicating the number of concurrent transactions.

1 to 500. Latency was measured as the delay between the moment the `peer chaincode invoke` command was issued and the time the transaction was committed to the ledger. The same command was used throughout the experiment, creating an academic credential with an incrementing credential ID and random values for the other attributes. For future work, the script could be modified to vary the commands, allowing different functions in the chaincode to be invoked. The results of this first evaluation are shown in Figure 2.

We also evaluated the throughput and latency for different transaction sizes by sending 100 transactions sequentially, without any concurrent execution. To simulate varying payload sizes, the same `peer chaincode invoke` command was issued repeatedly, each time with randomly generated arguments corresponding to payload sizes of 32 B, 64 B, 128 B, and 512 B. The goal was to observe whether increasing the transaction size would significantly impact network performance. The results of 5 experiments are presented in Figure 3.

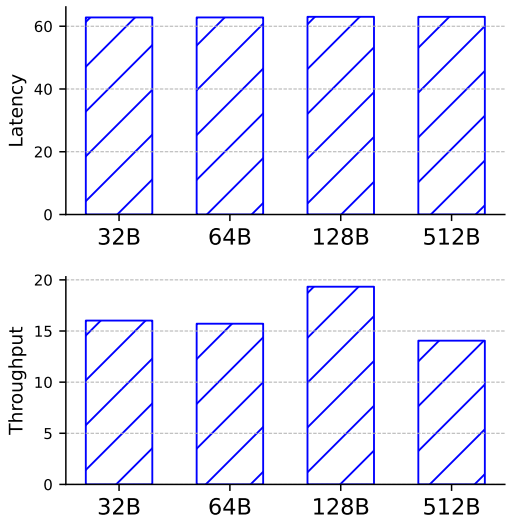

Figure 3: Latency (ms) and throughput (tx/s) of the network for different message sizes

4 Discussion

Figure 2 shows a typical throughput vs. latency curve, highlighting the tradeoff between the two. Indeed, when there are few users submitting transactions, the latency is at its lowest. Up to a limited number of concurrent transactions (such as 5), the throughput increases considerably while maintaining low latency. However, as the number of parallel transactions continues to grow, the throughput reaches a saturation point, whereas the latency rises significantly. Note that the median latency is used in this graph, and we found our values surprisingly low for such a network, which would typically exhibit latencies around 2–3 seconds. We hypothesize an error in the timing calculation that may have led to these unexpectedly low values. The actual latency values are not critical in this context, as the network runs in Docker containers, which do not accurately reflect the performance of a real production environment. Yet, the overall trend of the throughput vs. latency curve remains consistent with theoretical expectations, showing the characteristic performance tradeoff in blockchain-based systems.

Figure 3 shows that the latency remains constant regardless of the size of the transactions sent by the user. It also does not exhibit a clear or meaningful trend in throughput that would allow for further analysis. Therefore, we conclude that the performance of the Fabric network is largely unaffected by transaction size under the tested conditions—unless an error was made in the measurement script. This suggests that network overhead and consensus mechanisms dominate performance characteristics more than payload size in this context.

5 Conclusion

This project demonstrated the design and implementation of a blockchain-based system for secure talent credential verification using Hyperledger Fabric. By leveraging a permissioned network architecture, smart contracts, and a REST API interface, the system enables trusted interactions among talents, institutions, and companies. Credentials can be created, verified, and managed in a decentralized and tamper-resistant way, addressing many limitations of traditional verification systems.

The backend services were successfully integrated with a web-based frontend, providing users with a clear and intuitive interface for credential submission and validation. Performance evaluations were conducted to assess throughput and latency under various workloads, offering insight into the behavior of Fabric in different configurations.

One limitation encountered during development was the inability to deploy the full network as originally planned and depicted in Figure 1. Due to unresolved technical issues, the evaluation was instead conducted on a simplified version of the network, with fewer peers and a single orderer. While this still allowed for meaningful experimentation, a complete deployment would have provided more realistic performance data and additional insight into multi-organizational interaction. Another current limitation of the system is the absence of Private Data Collections, which would allow sensitive credential information to be shared selectively with authorized parties. Integrating this feature in future work will enhance privacy by allowing data to be shared only with authorized parties, while preserving the system's integrity.

6 Availability

The source code for this project can be found at https://github.com/benjaminvauchel/hyperledger-fabric.

References

[1] Kiratijuta Bhumichitr and Songsak Channarukul. Acachain: Academic credential attestation system using blockchain. In *Proceedings of the 11th International Conference on Advances in Information Technology*, IAIT '20, New York, NY, USA, 2020. Association for Computing Machinery.

[2] Marian Ileana and Angel Popgeorgiev. The use of blockchain technology for the validation and certification of competencies in education. In *Proceedings of the International Conference on Computer Systems and Technologies 2024*, CompSysTech '24, page 24–28, New York, NY, USA, 2024. Association for Computing Machinery.

[3] Atta ur Rehman Khan and Raja Wasim Ahmad. Blockchain-based academic degrees issuance and attestation. In *2022 International Conference on IT and Industrial Technologies (ICIT)*, pages 1–6, 2022.

[4] OpenCerts. Opencerts: An easy way to check and verify your certificates, 2020. Accessed: 2025-02-11.

[5] Philipp Schmidt. Blockcerts—an open infrastructure for academic credentials on the blockchain. *MIT Media Lab*, 2016.

[6] Aamna Tariq, Hina Binte Haq, and Syed Taha Ali. Cerberus: A blockchain-based accreditation and degree verification system. *IEEE Transactions on Computational Social Systems*, 10(4):1503–1514, 2023.