Cryptographic Requirements Of Verifiable Credentials For Digital Identification Documents

Tharunkumaar K
[1]Dept of Computer Applications
[2]Assistant professor, Dept of Computer Applications
[1, 2] Dr. M.G.R. Educational and Research Institute, Chennai, India

Abstract- *Digital identity verification is a critical requirement in modern e-governance and passport management systems. Traditional passport verification methods suffer from manual delays, identity fraud, and unauthorized access vulnerabilities. This paper presents a secure, multi-layer digital identity verification system that integrates Aadhaar QR code scanning, One Time Password (OTP) authentication, and FaceNet-based facial recognition to ensure tamper-resistant passport verification. The system is built using the Spring Boot framework for backend processing, MySQL for database management, and Python for FaceNet-based facial comparison. The proposed approach significantly reduces fake identity submissions, minimizes manual intervention, and improves authentication accuracy. Experimental results demonstrate that all core modules — including QR verification, OTP validation, and face matching — achieved 100% pass rates in functional testing. The system represents a scalable, secure framework suitable for e-governance and national identity management applications.*

Keywords: Verifiable Credentials, Aadhaar QR Verification, FaceNet, OTP Authentication, Passport Verification, Digital Identity, Spring Boot, Biometric Security.

I. INTRODUCTION

Digital transformation in government services has accelerated the need for robust identity verification frameworks. Conventional passport verification systems rely heavily on physical document examination and manual cross-referencing, processes that are time-consuming, error-prone, and susceptible to forgery. Identity fraud, duplicate passport generation, and unauthorized access to government services represent growing threats in the digital era.

The advent of cryptographic mechanisms and biometric authentication technologies offers a transformative approach to these challenges. Verifiable credentials, as defined by the W3C standard, provide a decentralized, tamper-evident method for representing identity claims. When combined with government-issued biometric data such as India's Aadhaar, these mechanisms offer a reliable foundation for secure digital identification.

This paper proposes a comprehensive passport verification system that leverages three core authentication layers: (1) Aadhaar QR code extraction and validation, (2) email-based OTP authentication, and (3) FaceNet deep learning facial recognition. The integration of these layers within a Spring Boot and Python-based architecture creates a reliable, scalable, and fraud-resistant identity verification platform. The system aims to reduce manual effort, improve verification accuracy, and support the broader goals of India's digital governance initiatives.

The remainder of this paper is structured as follows: Section II reviews related work; Section III describes the system architecture and methodology; Section IV presents the implementation details; Section V discusses experimental results and testing; Section VI concludes with directions for future work.

II. RELATED WORK

Several research efforts have explored digital identity verification using biometric and cryptographic approaches. Camenisch and Lysyanskaya [1] introduced anonymous credential systems that laid the theoretical groundwork for privacy-preserving identity verification. Their work demonstrated how users could prove attributes about themselves without revealing unnecessary personal data.

Schroff et al. [2] proposed FaceNet, a unified deep learning system for face recognition and verification that maps facial images directly to a compact Euclidean space. FaceNet achieves state-of-the-art performance on standard benchmarks, making it a strong candidate for real-time identity matching in high-security applications.

The W3C Verifiable Credentials Data Model [3] provides a standardized framework for expressing credentials in a cryptographically verifiable manner, suitable for integration with decentralized identity systems. Prior systems such as uPort and Sovrin have attempted to apply these standards, though adoption in government identity systems remains limited.

Bhattacharyya et al. [4] examined QR-code-based identity verification in the context of Aadhaar, noting that encrypted QR data provides a reliable offline mechanism for identity validation. Their analysis highlighted vulnerabilities in purely paper-based verification that digital extraction resolves.

Despite these advances, few systems integrate all three mechanisms — QR extraction, OTP validation, and biometric facial matching — into a unified, production-ready passport verification framework. This paper addresses that gap.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

A. Overall Architecture

The proposed system follows a three-tier client-server architecture. The presentation layer is built with HTML, CSS, and JavaScript for responsive user interaction. The application layer uses the Spring Boot framework to manage business logic, authentication, and API routing. The data layer employs MySQL for persistent storage of user records, application data, and verification results. Python serves as an auxiliary processing layer for FaceNet-based facial comparison via REST API calls from the Spring Boot backend.

Figure 1 conceptually illustrates the data flow: a user registers, submits a passport application, scans their Aadhaar QR code, completes OTP verification, and finally undergoes facial verification administered by an authorized officer. Each stage is gated by the success of the previous, forming a sequential multi-factor authentication chain.

B. Aadhaar QR Verification Module

The Aadhaar QR code encodes personal data — including name, Aadhaar number, date of birth, gender, address, and a photograph — in encrypted form. The system scans the QR code via a browser-integrated scanning interface and decodes the payload using established decryption protocols. The extracted data is then automatically compared against the details entered by the applicant in the passport application form. A mismatch at this stage terminates the verification workflow and logs a security event.

C. OTP Authentication Module

Upon successful Aadhaar verification, the system generates a cryptographically random six-digit OTP and dispatches it to the applicant's registered email address using SMTP. The OTP has a 60-second expiry window to mitigate replay attacks. The applicant must submit the correct OTP to proceed to facial verification. This layer protects against scenarios where an attacker possesses the physical Aadhaar card but not the associated email account.

D. FaceNet Facial Recognition Module

The final verification stage employs FaceNet, implemented in Python using the face_recognition library built on dlib's deep metric learning model. During verification, an authorized administrator captures a live webcam image of the applicant. The Python module computes 128-dimensional facial embeddings for both the Aadhaar photograph and the live capture, then calculates the Euclidean distance between them. A distance below the empirically determined threshold of 0.6 indicates a match and triggers passport approval. A failed match updates the application status to rejected and notifies the administrator.

E. Database Design

The MySQL schema comprises four primary entities: User (id, name, email, password, aadhaarNumber, address, image), PassportApplication (id, userId, fullname, dob, gender, aadhaarNo, phone, address, aadhaarDocument, status), OTPRecord (id, userId, otpCode, expiryTime, verified), and AdminLog (id, action, timestamp, userId). Referential integrity is enforced via foreign key constraints, and sensitive fields such as passwords are stored as BCrypt hashes.

IV. IMPLEMENTATION

A. Technology Stack

The backend is implemented in Java 17 using Spring Boot 3.x with Spring Security for session management and access control. MySQL 8.0 serves as the relational database, accessed via Spring Data JPA with Hibernate ORM. The Python 3.10 microservice exposes a Flask REST endpoint for facial comparison requests. The frontend uses Thymeleaf templates rendered server-side, styled with Bootstrap 5 for responsive design.

B. Key Algorithms

The secure credential validation algorithm hashes submitted passwords with BCrypt (cost factor 12) before comparison against stored hashes. Session tokens are generated using Java's SecureRandom and stored server-side with a 30-minute inactivity timeout. The identity matching algorithm performs field-level string comparison between

Aadhaar-extracted data and application form data, normalizing whitespace and case before comparison to minimize false negatives.

For facial recognition, FaceNet embeddings are computed using a pre-trained model (128-d output). The Euclidean distance d(e1, e2) between embedding vectors e1 and e2 determines match outcome: if $d < 0.6$, the identity is confirmed; otherwise it is rejected. This threshold was validated against the system's test dataset to minimize false acceptance and false rejection rates.

C. Security Measures

SQL injection is prevented through parameterized queries via JPA. Cross-Site Scripting (XSS) protection is provided by Thymeleaf's automatic output escaping. CSRF tokens are enforced on all state-changing POST requests via Spring Security. All communication between the browser and server occurs over HTTPS. Administrator actions are logged with timestamps and IP addresses to support audit trails.

V. EXPERIMENTAL RESULTS AND TESTING

A. Testing Methodology

The system underwent seven categories of testing: validation, integration, system, functional, performance, security, and user acceptance testing. All modules were evaluated against predefined test cases using both valid and invalid inputs to verify correct handling of edge cases and malicious inputs.

B. Test Results

Table I summarizes the key functional test cases and their outcomes. All seven primary test scenarios passed successfully, confirming the correctness of authentication workflows, QR parsing, OTP delivery, and facial matching.

TABLE I. FUNCTIONAL TEST RESULTS

Test ID	Description	Input	Expected	Result
TC01	User Login	Valid credentials	Login success	Pass
TC02	Invalid Login	Wrong password	Error message	Pass
TC03	Passport Apply	Valid details	Submitted	Pass
TC04	Aadhaar QR Verify	Valid QR code	Verified	Pass
TC05	OTP Auth	Valid OTP	OTP verified	Pass
TC06	Face Verification	Live face image	Face matched	Pass
TC07	Admin Approval	Verified request	Passport approved	Pass

C. Performance Observations

The Aadhaar QR decoding and field comparison process completes in under 500 ms under normal load. OTP email dispatch averages 1.2 seconds end-to-end. FaceNet embedding computation and comparison requires approximately 800 ms per verification request on a standard Intel Core i5 machine with 8 GB RAM. The Spring Boot backend maintained stable response times under simulated concurrent load of 50 simultaneous users, with no session collisions or data corruption observed.

D. Security Testing

SQL injection attempts using common payloads were blocked by JPA parameterized queries. Attempts to reuse expired OTPs returned rejection responses correctly. Submitting a mismatched face image (different person) consistently produced Euclidean distances above the 0.6 threshold, triggering rejection. Brute-force login attempts were mitigated by Spring Security's account lockout mechanism after five consecutive failures.

VI. CONCLUSION AND FUTURE WORK

This paper presented a secure, multi-factor digital passport verification system integrating Aadhaar QR authentication, OTP validation, and FaceNet facial recognition. The system addresses significant shortcomings in traditional manual verification processes by automating identity validation through cryptographic and biometric mechanisms. Comprehensive testing confirmed that all functional modules operate correctly, and performance benchmarks indicate suitability for real-world government deployment.

Future work will explore integration with the DigiLocker API for direct government document retrieval, blockchain-based immutable audit trails for verification records, and advanced anti-spoofing measures for the facial recognition module. Extending the system to support fingerprint and iris recognition would further strengthen the biometric factor..

REFERENCES

[1] J. Camenisch and A. Lysyanskaya, "An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation," in Proc. EUROCRYPT, 2001, pp. 93–118.

[2] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in Proc. IEEE CVPR, 2015, pp. 815–823.
[3] M. Sporny, D. Longley, and D. Chadwick, "Verifiable Credentials Data Model 1.0," W3C Recommendation, Nov. 2019. [Online]. Available: https://www.w3.org/TR/vc-data-model/
[4] S. Bhattacharyya, A. Bhattacharya, and S. Nandi, "QR Code-Based Offline Aadhaar Identity Verification," in Proc. IEEE ICACCI, 2018, pp. 1123–1129.
[5] Spring Boot Official Documentation, Version 3.x, Pivotal Software, 2024. [Online]. Available: https://spring.io/projects/spring-boot
[6] Python Software Foundation, "Python 3.10 Documentation," 2024. [Online]. Available: https://www.python.org/doc/
[7] MySQL 8.0 Reference Manual, Oracle Corporation, 2024. [Online]. Available: https://dev.mysql.com/doc/
[8] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd ed. Sebastopol, CA: O'Reilly Media, 2019.