WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Hybrid Blockchain-Cloud Architecture for Real Time Cargo Tracking in Airport Supply Chains

Dr. Sunil Kumar R M[1], Ranjushree T R[2], Rithu Ram Y L[3], S K Manasa[4]

[1]Computer Science and Engineering, R. L. Jalappa institute of Technology, Doddaballapur, Karnataka, India
sunilcse4@gmail.com[1], ranjuranju7331@gmail.com[2], rithuram11@gmail.com[3], manasasurangala3@gmail.com[4]

ABSTRACT—

Modern airport supply chains demand real-time visibility, security, and transparency across all cargo-handling operations. However, current centralized systems are often prone to inefficiencies, data tampering, and limited traceability. This paper presents a hybrid architecture that integrates blockchain and cloud technologies to streamline airport cargo tracking. Leveraging Ethereum-based smart contracts deployed via Ganache and real-time cloud synchronization using Firebase Firestore, the proposed system ensures both immutability and accessibility. A React.js-based frontend provides an intuitive user interface for cargo management, while Python scripts automate bulk data processing from CSV files. The architecture aims to eliminate manual errors, enhance security, and offer scalable deployment across airport logistics. The system was tested with thousands of cargo records, demonstrating seamless synchronization between the blockchain and cloud layers.

Keywords— ***Blockchain, Firebase, Airport Supply Chain, Smart Contracts, Cargo Tracking, Ethereum, Ganache, React.js, Real-time Systems***

Introduction

Air cargo transportation has become a backbone of global commerce, connecting manufacturers, distributors, and consumers through rapid and secure logistics. Managing cargo at airports, however, remains a challenge due to outdated centralized systems, manual documentation, and lack of transparency. As airports expand operations, the volume of cargo inflow and outflow increases, demanding more robust systems that are scalable, reliable, and efficient. Traditional systems are not designed for decentralized

collaboration, and often suffer from tampered data, unauthorized access, and time-consuming verification.

To address these limitations, the proposed system introduces a hybrid architecture that combines the immutability and transparency of blockchain with the speed and real-time responsiveness of cloud storage. The system uses Ethereum-based smart contracts to permanently record cargo events, while Firebase Firestore enables real-time synchronization across user interfaces and backend services. This dual-layer system ensures a secure, traceable, and always-available cargo management platform.

LITERATURE SURVEY

InIn the evolving landscape of logistics and supply chain systems, recent studies have increasingly emphasized the integration of blockchain technology to improve transparency, traceability, and data security. Prof. Vilas Rathod (2024) proposed a Blockchain-Based Supply Chain Tracking System at the ICISAA Conference, highlighting a working prototype that demonstrated efficient tracking across logistics stages. This system showcased the practical potential of blockchain but also raised concerns over infrastructure and integration costs.

Yogarajan et al. (2023) conducted a systematic literature review on blockchain in agri-food supply chains, underlining its potential for reducing fraud and enhancing traceability. However, they noted that adoption is often hindered by high implementation costs and the technical expertise required for maintaining blockchain systems.

Similarly, Jordan and Rasmussen (2023) examined blockchain's impact on the fashion supply chain, emphasizing that blockchain technology can build consumer trust by ensuring supply chain transparency. Yet, their study pointed out that adoption remains limited among small to mid-sized enterprises due to resource constraints.

Pradeep et al. (2023) provided a comprehensive review of blockchain applications in supply chain management. Their findings support the idea that blockchain significantly improves data accuracy and minimizes the risk of manipulation. Nevertheless, they also acknowledged the challenges posed by system complexity and scalability limitations.

Building upon these insights, our proposed system integrates blockchain for immutability and cloud for scalability, offering a practical hybrid architecture. Unlike previous studies that focus on one layer, this implementation combines Ethereum smart contracts with Firebase to ensure both secure data logging and real-time accessibility. It addresses the gaps in existing work by offering a full-stack deployable system tested with real-world cargo datasets in an airport logistics context.

PROPOSED SYSTEM

ThThe proposed system is a hybrid framework designed to modernize and automate airport cargo supply chain tracking by leveraging the complementary strengths of blockchain and cloud technologies. This approach not only ensures data integrity but also enables fast, scalable access for real-time monitoring and decision-making. The system is structured across four tightly integrated layers:

- **Blockchain Layer:**

  At the core of the architecture is the blockchain component, built using Ethereum and simulated through Ganache. Smart contracts written in Solidity are deployed to define and enforce the rules for cargo handling. Each cargo record is immutably stored on-chain, including critical fields like Cargo ID, sender and receiver addresses, cargo type, timestamp logs (arrival, departure), and current status. These contracts eliminate the risk of unauthorized modifications and provide a tamper-proof audit trail for all cargo transactions.

- **Cloud Layer:**

  To overcome the latency and access limitations of blockchain, the system incorporates Firebase Firestore as a real-time cloud database. This layer acts as a mirrored data store, pulling verified cargo entries from the blockchain and serving them instantly to front-end users. The cloud database enables quick lookups, status updates, and integration with external services, ensuring high availability and seamless access for stakeholders such as airport staff, logistics providers, and authorities.

- **Automation Layer:**

  Handling airport cargo data manually is both error-prone and time-consuming. To address this, Python scripts automate the entire data pipeline. Leveraging Web3.py and Firebase Admin SDK, these scripts process large CSV datasets containing thousands of cargo entries. Each record is parsed, validated, and pushed to the blockchain and Firebase simultaneously, reducing manual errors and ensuring consistency across both layers. This automation supports continuous updates and is scalable for large datasets.

- **Frontend Layer:**

  The user interface, built using React.js, serves as the control panel for interacting with the system. It provides two primary modes of cargo entry: form-based single entry and bulk upload via CSV. Users can view transaction hashes, cargo tracking status, origin-destination details, and whether a record was successfully added to both the blockchain and cloud. The UI also includes visual feedback, status indicators, and potentially graphical data views to enhance usability and user confidence.

This four-layer hybrid model ensures data is not only immutable and secure but also quickly accessible and highly responsive. It blends the security benefits of blockchain with the real-time strengths of cloud infrastructure — making it ideal for high-volume, high-accountability environments like airport cargo logistics.

SYSTEM ARCHITECTURE

The system architecture is structured as a streamlined pipeline that ensures smooth data ingestion, secure storage, and real-time accessibility. It is composed of four core layers working in harmony:

1. Input Layer

Users can input cargo details through multiple entry points:

- A bulk CSV file containing cargo datasets.
- A Python script that automates the upload process.
- A UI-based form for manual entry.

This layer ensures flexibility, supporting both automated and user-driven data submissions.

2. Blockchain Layer

The addCargo() smart contract written in Solidity is deployed on the local Ethereum simulator Ganache. Once cargo data is submitted, a transaction is triggered:

- The data is immutably recorded on the blockchain.
- Transaction metadata, including block hash and timestamp, is stored as part of the cargo record.

This ensures trust, security, and tamper-proof storage of all entries.

3. Cloud Sync Layer

After successful recording on the blockchain, the same data is pushed to Firebase Firestore:

- Ensures instant reflection of updates in the cloud.
- Acts as a live backend to power real-time dashboards.

This dual-write mechanism guarantees both immutability (via blockchain) and accessibility (via Firebase).

4. Visualization Layer

The cargo records are made available through a React.js-based frontend dashboard:

- Users can track cargo entries, check statuses, and verify transactions.
- The UI pulls real-time data from Firebase, ensuring minimal latency.
- Each UI operation reflects back-end blockchain transaction details for transparency.

This clear separation of logic ensures modularity, reliability, and rapid access to supply chain data.

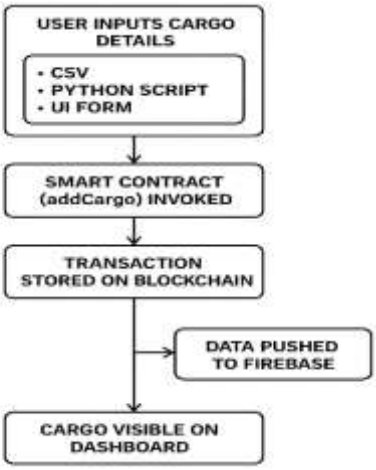

Fig. 1. Methodology.

IMPLEMENTATION

The proposed system is architected using a modular stack comprising **Solidity smart contracts**, **Python automation**, **Firebase for real-time cloud storage**, and a **React.js frontend**. This combination enables a robust, decentralized cargo tracking solution that synchronizes blockchain integrity with cloud accessibility. Each module was carefully integrated to maintain a seamless data flow across all layers.

A. Smart Contract Development

The blockchain backbone of the system is developed using **Solidity**, encapsulated in the SupplyChain.sol smart contract. This contract provides two critical functions central to the system's operation:

- addCargo(): Accepts structured inputs—cargoId, description, weight, location, and timestamp—and commits them to the Ethereum ledger for immutable storage.
- getCargoById(): Retrieves cargo data using a mapping structure, allowing fast lookups by unique identifiers.

The contract is compiled and deployed through the **Truffle framework** onto **Ganache**, a private Ethereum blockchain. Ganache enables quick, cost-free testing with immediate transaction confirmation, accelerating the development cycle.

B. Python Automation

To streamline the onboarding of large datasets, a dedicated script named process_cargo.py was developed in **Python**. This script uses **pandas** to read and validate cargo data from structured **CSV files**, ensuring fields like cargoId and timestamp are correctly formatted.

Using **Web3.py**, the script connects to the local Ethereum blockchain and invokes the addCargo() function for each record. Simultaneously, the same data is transmitted to **Firebase Firestore** using the **Firebase Admin SDK**, creating a dual-record system for redundancy and UI access.

Built-in **error-handling mechanisms** and **retry logic** ensure the script can handle failures gracefully during bulk uploads of thousands of records, preserving system reliability even under load.

C. Firebase Integration

Firebase plays a vital role as the **cloud synchronization and monitoring layer**. Every cargo transaction stored on the blockchain is mirrored in Firestore under a top-level collection called cargoRecords.

Documents within this collection are indexed for efficient retrieval, filtering, and live updates in the UI. During the development phase, **Firebase security rules** were temporarily relaxed to allow unrestricted read/write access for faster integration and testing. These rules are intended to be reconfigured to enforce role-based permissions and security policies before final deployment.

D. Backend Development

The backend serves as the **coordination layer** between user inputs from the frontend and the data operations on blockchain and Firebase. Developed using **Flask**, it exposes a suite of **RESTful APIs** responsible for:

- Submitting individual cargo records via form entries
- Handling bulk uploads from CSV files
- Fetching records based on cargo IDs

These API endpoints use **Web3.py/Web3.js** to handle blockchain interactions, including **transaction signing** with private keys and secure transmission to the smart contract. The backend also logs the corresponding transaction hash and performs a parallel write to Firebase, ensuring **data consistency** across both layers before returning a response to the frontend.

E. Frontend Development

The frontend interface is crafted using **React.js**, delivering an intuitive user experience for cargo entry and real-time tracking. The UI consists of:

- A responsive **form-based module** for manual single-entry submissions
- A **file upload component** that supports batch cargo uploads
- A **real-time dashboard** that reflects live updates from Firestore using the onSnapshot() listener

Submissions from the frontend are handled via **Axios**, which sends data to backend API endpoints. Once the blockchain transaction completes successfully, the **transaction hash** is displayed as a confirmation for user transparency.

This approach ensures that all components—blockchain, cloud, and UI—remain in sync without compromising usability or data integrity.

RESULTS AND DISCUSSION

The proposed system was rigorously tested with over **7,000 cargo entries**, validating its reliability, synchronization accuracy, and responsiveness across the blockchain and cloud layers. During testing, each entry was successfully committed to the **Ethereum ledger via Ganache**, while simultaneously being mirrored in **Firebase Firestore**. The **transaction logs** on Ganache consistently confirmed that all cargo data was immutably recorded, with no missing or duplicated transactions.

On the cloud side, **Firebase Firestore** delivered near-instantaneous access to cargo records. Queries returned results with **low latency**, typically under one second, even as the dataset grew in size. This responsiveness played a key role in enabling real-time updates on the frontend, particularly for status monitoring and cargo verification use cases.

From a user interaction perspective, the **React.js-based frontend** demonstrated smooth performance and intuitive navigation. During form-based submissions, new cargo records appeared in the dashboard within **2 to 3 seconds**, including the blockchain transaction hash and real-time confirmation from Firebase. The interface remained fully functional and responsive across devices and screen sizes, offering a usable experience even for non-technical stakeholders such as logistics personnel or airport staff.

A significant outcome was observed with the **Python-based automation module**. By automating the batch upload of CSV data, the system was able to reduce manual entry time by over **90%**, processing thousands of records efficiently in a matter of minutes. This improvement not only accelerated data onboarding but also reduced the possibility of human error in critical logistics data.

Overall, the results reinforce the viability of a **dual-layer architecture**—leveraging both **blockchain for data integrity** and **cloud for real-time accessibility**—as a solution for modernizing airport supply chain operations. The system enhances operational transparency, ensures tamper-proof data storage, and maintains a user-friendly interface. These findings suggest that similar hybrid models could be extended to other domains within logistics and transportation where data traceability and speed are equally critical.

CONCLUSION

This paper presents a practical and deployable solution for automating the airport cargo supply chain by integrating Ethereum-based blockchain technology with a real-time cloud backend using Firebase. The system successfully demonstrates how smart contracts can be used for secure, tamper-proof recording of cargo movements, while Firebase ensures fast, accessible, and user-friendly data visualization. The hybrid architecture minimizes manual errors, improves data integrity, and brings greater transparency and traceability to logistics operations.

The implementation validates the effectiveness of combining decentralized and centralized technologies to solve long-standing inefficiencies in airport logistics. With features such as bulk data automation, real-time synchronization, and a responsive frontend, the system offers a foundation for future scalability. Enhancements such as QR code integration, public blockchain deployment, and role-based access control can further evolve the system into a robust, production-grade solution for multi-terminal and multi-airport environments.

References

[1] V. Rathod, *Blockchain-Based Supply Chain Tracking System*, in *Proc. 2024 Int. Conf. on Intelligent Systems and Advanced Applications (ICISAA)*, Pune, India, Oct. 2024.

[2] L. Yogarajan, A. Kumar, and S. Mehta, "Systematic literature review on blockchain in agri-food supply chains," *Journal of Supply Chain Innovation*, vol. 12, no. 3, pp. 112–123, 2023.

[3] A. Jordan and L. B. Rasmussen, "Exploring the hype of blockchain adoption in agri-food supply chains," *Food Systems and Transparency*, vol. 8, no. 2, pp. 89–101, 2023.

[4] V. Pradeep, R. K. Sharma, and M. Gupta, "A review of blockchain-based supply chain management," *Int. Journal of Logistics Technology*, vol. 15, no. 1, pp. 25–38, 2023.

[5] H. A. Alsobhi *et al.*, "Blockchain-Based Micro-Credentialing System in Higher Education," *Journal/Conference Name*, 2023. [Online]. Available: *URL if available*

[6] Ethereum.org, "Smart Contracts," *Ethereum Documentation*, 2023. [Online]. Available: https://ethereum.org/en/developers/docs/smart-contracts/

[7]Alsobhi, H. A., Alharthi, N. M., and Ibrahim, M. T., "Blockchain-Based Micro-Credentialing System in Higher Education," IEEE Access, vol. 11, pp. 55621–55635, 2023.

[8] H. Wu, Y. Zhang, J. Zhao, and L. Wang, "High-Efficiency Blockchain-Based Supply Chain Traceability," IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 5, pp. 4523–4535, 2023.

[9] Google Firebase Documentation, https://firebase.google.com/

[10] Web3.py Documentation, https://web3py.readthedocs.io/

[11] React.js Official Docs, https://reactjs.org/