International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Fake Product Identification Using Blockchain and QR Code

[Aarti Murkunde][1], [Suvarna Aute][2], [Ashwini Bhavle][3], [Akanksha Dukare][4], Prof. [Medhe S.S][5] ,HOD.[Dr.H.B.Jadhav][6]

[1]Student, Department of Computer Engineering, Adsul's Technical Campus, Chas, Ahilyanagar, India, [aartimurkunde@gmail.com]
[2]Student, Department of Computer Engineering, Adsul's Technical Campus, Chas, Ahilyanagar, India, [suvarnaaute4@gmail.com]
[3]Student, Department of Computer Engineering, Adsul's Technical Campus, Chas, Ahilyanagar, India, [ashwinibhavle4@gmail.com]
[4]Student, Department of Computer Engineering, Adsul's Technical Campus, Chas, Ahilyanagar, India, [dukareakanksha10@gmail.com]
[5]Guide, Department of Computer Engineering, Adsul's Technical Campus, Chas, Ahilyanagar, India, [suyogmedhe@gmail.com]
[6] HOD, Department of Computer Engineering, Adsul's Technical Campus, Chas, Ahilyanagar, India, [hem3577@gmail.com]

Abstract - Counterfeit goods cause financial loss to brands and health and safety risks to consumers, and the usual protections such as holograms, printed labels and plain barcodes are easy to copy. This paper presents a system that identifies fake products using the Ethereum blockchain and QR codes. When a manufacturer registers a product, its ID, name, batch, manufacturing date and manufacturer address are hashed with Keccak-256, and a Solidity smart contract stores the product ID, the hash, the manufacturer address and the registration time. Only approved manufacturer accounts can write to the contract. A QR code holding the product ID and the hash is printed on the product. When a consumer scans the code, the verification service reads the on-chain record and compares the hashes, so unknown or tampered codes are reported as Counterfeit. Because a valid code can still be photocopied onto fake items, every scan is logged with its time and location, and a code that is scanned too many times or at places too far apart for the time between scans is reported as Suspicious. The contract is developed with Hardhat, tested on Ganache and deployed on the Sepolia test network; the backend uses Node.js, Express and Web3.js with MongoDB, and the clients are a React portal and a React Native scanner app. The system is evaluated with genuine, fake, tampered and copied codes against a central database lookup and a blockchain lookup without scan logs, with a target scan-to-result time under 2 seconds.

Key Words: blockchain, Ethereum, smart contract, QR code, counterfeit detection, product authentication.

1. INTRODUCTION

Counterfeit products are found in almost every market: medicines, chargers and spare parts, cosmetics, food items and luxury goods. A fake medicine can contain the wrong dose or no active ingredient, and a fake battery can overheat. Apart from the risk to buyers, brands lose sales and reputation and governments lose tax revenue. In most cases the buyer has only the packaging to judge by, and the packaging is the easiest part of a product to copy.

The protections in common use do not solve this. Holograms and special printed labels add cost to each item but are copied by counterfeiters. Barcodes and simple QR codes only point to a product page, so the same code printed on a thousand fake boxes looks valid every time. Some companies keep a central database of serial numbers, but the database is controlled by one party, its records can be changed by anyone with access, and it stops working when its one server is down. None of these methods tells a buyer that the code in their hand was scanned in another city an hour earlier.

A blockchain is a ledger that is copied to many nodes, in which each block holds the hash of the previous block, so a stored record cannot be changed without the change being visible [1]. Ethereum adds smart contracts, programs that run on the chain and enforce rules such as who may write a record [2], [3]. Several student and research projects have linked product records on a blockchain to QR codes so that buyers can check a product by scanning it [4]–[8]. These systems remove the need to trust the seller, but most of them only check whether the scanned code exists on the chain. A counterfeiter who copies one valid code onto many fake items passes this check every time.

This work builds a complete verification system around this gap. The main contributions are: (i) a ProductRegistry smart contract that stores only a Keccak-256 hash and a few fixed-size fields per product, written only by approved manufacturer accounts; (ii) a registration flow that hashes the product details and generates a QR code holding the product ID and the hash; (iii) a verification flow that compares the scanned hash with the on-chain hash and checks a scan log for impossible travel and excess scans, giving the verdict Genuine, Counterfeit or Suspicious; and (iv) a test protocol with genuine, fake, tampered and copied codes that compares the system with a central database lookup and a blockchain lookup without scan logs.

The rest of the paper is organised as follows. Sec. 2 reviews related work. Sec. 3 describes the proposed system and the verification algorithm. Sec. 4 gives the experimental setup, Sec. 5 presents the results and discussion, and Sec. 6 concludes the paper.

2. RELATED WORK

2.1 Blockchain and Smart Contracts

Nakamoto [1] introduced a chain of blocks in which each block stores the hash of the previous one, with proof of work making it very costly to rewrite history; any change to an old record changes all later hashes and is easy to detect. Wood [2]

defined Ethereum, a blockchain with a virtual machine that runs smart contracts and charges gas for every operation. Christidis and Devetsikiotis [3] explained how smart contracts let parties that do not trust each other share a process without a middleman, and discussed where transaction cost and delay are a problem. Kshetri [9] described how blockchain improves data integrity and removes single points of failure, which is the weakness of a central serial number database.

2.2 Blockchain for Supply Chains

Tian [10] combined blockchain and IoT to trace food products through production, processing and distribution, with each stage writing its records to the ledger. Casino et al. [11] classified blockchain applications by sector and found supply chain tracking and provenance to be among the main use cases, while noting the cost of storing data on chain. Yli-Huumo et al. [12] listed throughput, latency and chain size as open problems. These limits suggest that a verification system should write to the chain rarely and store as little as possible there.

2.3 Blockchain-based Fake Product Detection

Gandhimathi et al. [4] stored product data on a blockchain and verified it by scanning a unique QR code. Reddy et al. [5] used a blockchain ledger and QR scanning so that customers can check authenticity, and Pansare et al. [6] removed the dependency on the store by using a decentralised ledger and QR code. Srikrishna Shastri et al. [7] applied blockchain with QR validation and suggested IoT devices for automatic detection. Darshan and Aadil [8] built a DApp that records and shows the ownership history of a product, which exposes items with an unknown owner but requires every party in the chain to record each transfer. Most of these systems verify that a code exists on the chain; comparatively little work uses the scan history of a code to detect copies, which is the focus of the present work.

3. PROPOSED SYSTEM

3.1 System Architecture

Fig. 1 shows the overall architecture. The system has four layers: registration, the blockchain layer, verification and the application layer. The ProductRegistry contract is written in Solidity, compiled and tested with Hardhat, and deployed on Ganache for development and on the Sepolia test network for demonstration. The application follows the MERN stack: MongoDB stores users, off-chain product details, scans and alerts, a Node.js and Express server exposes REST endpoints, and React is used for the manufacturer portal. The scanner app is built in React Native. The backend talks to the contract through Web3.js, so consumers do not need a crypto wallet or ether to verify a product.

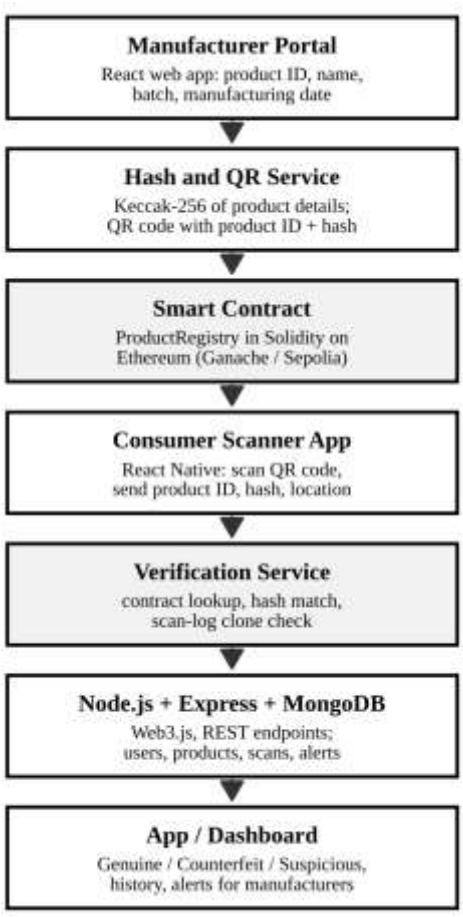


Fig -1: System architecture of the proposed fake product identification system

3.2 Product Registration and Data Storage

Only an account that the admin has approved can register products. The manufacturer enters the product ID, name, batch number and manufacturing date in the portal. The backend checks the fields and computes the product hash over the ABI-encoded fields and the manufacturer address m, using the Keccak-256 function [13], the variant of SHA-3 [14] used by Ethereum:

$$h = \text{Keccak-256}(\text{pid} \parallel \text{name} \parallel \text{batch} \parallel \text{date} \parallel m) \quad (1)$$

The contract is then called with (pid, h) from the manufacturer account. It rejects a sender that is not approved and a product ID that already exists, and stores the record shown in Table 1. Only fixed-size fields are kept on chain, because each new storage word costs gas; the readable details, the product image and the scan log are kept in MongoDB. A batch call registers many products in one transaction, which spreads the fixed transaction cost. After the transaction is confirmed, a QR code [15] holding the payload {pid, h, v} is generated at error correction level M and printed on the label. Fig. 2 shows the registration flow.

Table -1: Data stored for each product

Field	Type	Stored in	Purpose
productId	bytes32	Contract, QR	Key of the record
dataHash	bytes32	Contract, QR	Integrity check
manufacturer	address	Contract	Who registered it
registeredAt	uint256	Contract	Time of registration
Name, batch, date, image	text, file	MongoDB	Display to user

Field	Type	Stored in	Purpose
Scan log	documents	MongoDB	Clone check, history

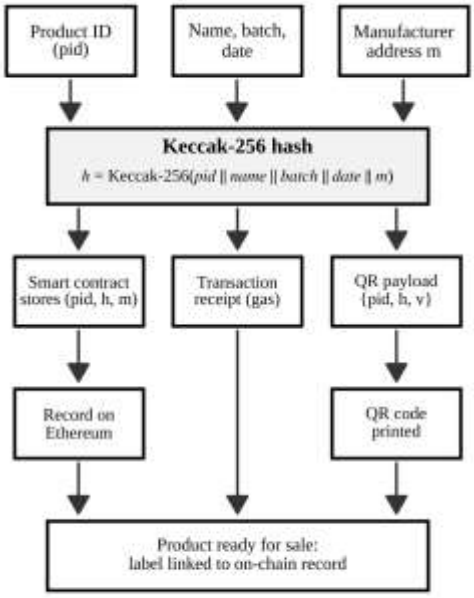


Fig -2: Product registration and QR code generation

3.3 Verification Algorithm

When a consumer scans a code, the app decodes the payload and sends the product ID, the scanned hash h′, the time τ and, if the user allows it, the location (φ, λ) to the /verify endpoint. The backend calls the read-only function getProduct(pid), which costs no gas and does not wait for a block, and evaluates

$$\text{match} = [R(\text{pid}) \text{ exists}] \wedge [R(\text{pid}).h = h'] \quad (2)$$

where R(pid) is the stored record. A code whose ID is unknown, whose record is revoked, or whose hash differs from the stored hash gives match = 0 and the verdict Counterfeit. Because the hash covers every printed field, changing even one character of the name or batch on a fake label makes the check fail. The probability that two different products get the same 256-bit hash is negligible.

3.4 Clone Detection from Scan Logs

A code that passes the hash check may still be a photocopy of a genuine code. Every scan is therefore stored in the scans collection, and the new scan is compared with the earlier scans of the same code. The distance between two scan locations is computed with the Haversine formula, with r = 6371 km:

$$d = 2r \cdot \arcsin \sqrt{[\sin^2(\Delta\varphi/2) + \cos\varphi_1 \cos\varphi_2 \sin^2(\Delta\lambda/2)]} \quad (3)$$

and the travel speed implied by the two scans is

$$v = d / (\tau_2 - \tau_1) \quad (4)$$

The clone flag is set when v is above v_{max} = 900 km/h, roughly the speed of a passenger aircraft, or when the total number of scans N is above a limit N_{max} set by the manufacturer for the product type. A product with match = 1 and the flag set is reported as Suspicious and an alert is raised on the manufacturer dashboard; otherwise it is reported as Genuine with its details and history. The decision is shown in Fig. 3. If the user denies location access, only the scan count is checked.

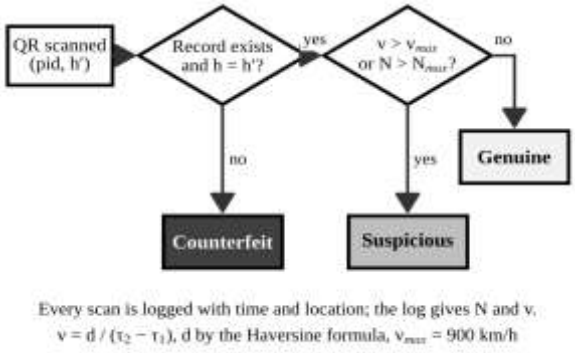


Fig -3: Verification decision for one scanned QR code

3.5 Baseline Methods

Two baselines are implemented on the same test data. **Central database lookup** prints a serial number in the QR code and checks it against a MongoDB table owned by the manufacturer; it reports Genuine whenever the serial exists. **Blockchain lookup only** follows the approach of [4]–[7]: it uses the same contract and hash check as the proposed system but has no scan log, so it reports Genuine whenever match = 1. The proposed system adds the clone check of Sec. 3.4 to the second baseline, so the difference between them shows the effect of the scan log alone.

3.6 Web Portal, Scanner App and Dashboard

The manufacturer portal lets an approved user register single products or batches, view the transaction hash of each registration, and download a printable sheet of QR codes. The scanner app opens the camera, reads the code and shows the verdict as one large word together with the product name, batch, manufacturer and date, and the number and places of earlier scans. Consumers do not need to log in to scan. The dashboard shows scan counts by product and region and lists Suspicious codes with the scans that triggered them, so the manufacturer can trace the shop or area where copies are being sold.

4. EXPERIMENTAL SETUP

4.1 Test Protocol

A set of products is registered on the Sepolia test network, and three kinds of test codes are prepared: genuine codes printed from the registered records; fake codes with product IDs that were never registered, together with tampered codes in which the name, batch or date of a registered product is changed before printing (both are grouped as Fake in Table 2); and copied codes in which one genuine code is printed several times and scanned at places and times that a single item could not reach, or more often than N_{max}. Scan locations and times for the copied codes are set through the test client so that the cases can be repeated exactly. Every test code is scanned with each of the three methods and the verdicts are compared with the known label.

4.2 Evaluation Metrics

Detection accuracy over K test scans, where C is the number of scans that get the correct verdict, is:

$$\text{Accuracy} = (C / K) \times 100\% \quad (5)$$

The mean scan-to-result time, from the moment the code is captured to the moment the verdict is shown, is:

$$T = (1/K) \Sigma_k (\tau_{res,k} - \tau_{scan,k}) \quad (6)$$

The cost of a registration transaction is computed from the gas reported in the transaction receipt:

$$\text{Cost} = \text{gasUsed} \times \text{gasPrice} \quad (7)$$

Gas is reported both for a single registration and per product for a batch registration, since the gas used does not depend on the market price of ether.

4.3 Implementation Details

The contract is written in Solidity 0.8 and tested with Hardhat on a local Ganache network before deployment on Sepolia through an RPC provider. The backend uses Node.js 20, Express 4, Web3.js 4 and MongoDB 7, with bcrypt for password hashing and JSON Web Tokens for protected routes. QR codes are generated with the qrcode library. The portal uses React 18 and the scanner app React Native. Response times are measured on an Android phone over a normal mobile data connection, and every method is tested with the same set of codes.

5. RESULTS AND DISCUSSION

Table 2 gives the detection accuracy and mean scan-to-result time of the three methods for each kind of test code. Higher accuracy and lower time are better.

Table -2: Detection results on the test codes (time in seconds)

Test case	Method	Correct	Accuracy (%)	Time (s)
Genuine	Central database lookup	[ ]	[ ]	[ ]
	Blockchain lookup only	[ ]	[ ]	[ ]
	Proposed (with scan log)	[ ]	[ ]	[ ]
Fake	Central database lookup	[ ]	[ ]	[ ]
	Blockchain lookup only	[ ]	[ ]	[ ]
	Proposed (with scan log)	[ ]	[ ]	[ ]
Copied	Central database lookup	[ ]	[ ]	[ ]
	Blockchain lookup only	[ ]	[ ]	[ ]
	Proposed (with scan log)	[ ]	[ ]	[ ]

[Fill Table 2 with the values obtained from the test runs and describe them here: the accuracy of each method for each kind of test code, the mean scan-to-result time, and the gas used for a single registration and per product in a batch registration on Sepolia.]

Some behaviour can be predicted from the design independent of the measured values. The central database lookup cannot detect a tampered label if the attacker keeps a valid serial number, and it accepts every copy of a genuine code. The blockchain lookup rejects fake and tampered codes because the hash no longer matches, but like the database it accepts copied codes. Only the proposed method can flag copies, and only after a copy has been scanned: if a buyer scans a copied code before the genuine item is ever scanned, the first scan looks normal and the copy is caught at a later scan. Scans from the same city at short intervals, such as a shop owner checking stock, should not trigger the speed check, which is why N_{max} is set per product type. Registration has to wait for one block (about 12 s on Sepolia), but verification only reads the contract, so its time is set mainly by the network round trip.

6. CONCLUSION

This paper presented a system that identifies fake products by linking a unique QR code on each product to a record on the Ethereum blockchain. Only approved manufacturers can register products, only a Keccak-256 hash and a few fixed-size fields are stored on chain, and any change to the printed product data makes verification fail. A scan log with an impossible-travel check and a scan limit lets the system flag copied codes, which a plain blockchain lookup accepts. Consumers verify a product with a phone camera and need no wallet or ether. Future work includes NFC and IoT tags that are harder to copy than printed codes, machine learning on scan logs to find unusual scan patterns, and full supply chain tracking in which each transfer between parties is recorded on the chain.

ACKNOWLEDGEMENT

We thank our guide, Prof. [Guide Name], for guidance throughout this work, and the Head of the Department of Computer Engineering, Adsul's Technical Campus, Chas, Ahilyanagar, for providing the facilities needed for the project.

REFERENCES

1. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.
2. G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," Ethereum Yellow Paper, 2014.
3. K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," IEEE Access, vol. 4, pp. 2292–2303, 2016.
4. K. Gandhimathi et al., "Fake Product Identification using Blockchain and QR Code," 2025.
5. G. S. S. Reddy et al., "Fake Product Detection Using Blockchain Technology," 2023.
6. J. R. Pansare et al., "Fake Product Identification System Using Blockchain," 2023.
7. C. Srikrishna Shastri et al., "Fake Product Detection Using Blockchain Technology," 2022.
8. N. Darshan and M. Aadil, "Identification of Counterfeit Product Using Blockchain," 2023.
9. N. Kshetri, "Blockchain's Roles in Strengthening Cybersecurity and Protecting Privacy," Telecommunications Policy, vol. 41, no. 10, pp. 1027–1038, 2017.
10. F. Tian, "A Supply Chain Traceability System for Food Safety Based on HACCP, Blockchain & Internet of Things," in Proc. IEEE Int. Conf. on Service Systems and Service Management (ICSSSM), 2017.
11. F. Casino, T. K. Dasaklis and C. Patsakis, "A Systematic Literature Review of Blockchain-based Applications: Current Status, Classification and Open Issues," Telematics and Informatics, vol. 36, pp. 55–81, 2019.
12. J. Yli-Huumo, D. Ko, S. Choi, S. Park and K. Smolander, "Where Is Current Research on Blockchain Technology? A Systematic Review," PLoS ONE, vol. 11, no. 10, e0163477, 2016.
13. G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, "The Keccak Reference," version 3.0, 2011.
14. National Institute of Standards and Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," FIPS PUB 202, 2015.
15. ISO/IEC 18004:2015, "Information Technology: Automatic Identification and Data Capture Techniques: QR Code Bar Code Symbology Specification," 2015.