

# TraceGuard: A QR Code and Blockchain-Based Framework for Counterfeit Product Detection and Supply Chain Transparency

Abhishek Gayake, Siddhesh Bokefod

Department of Computer Engineering, Pune, Maharashtra, India

## ARTICLE INFO

### Article History:

Accepted : 06 August 2026

Published : 08 August 2026

### Publication Issue

Volume 01, Issue 03

July-September 2026

### Page Number

25-30

## ABSTRACT

Counterfeiting causes measurable revenue loss and erodes consumer trust across manufacturing sectors, yet most anti-counterfeiting deployments still depend on centralized product registries that a single administrator can alter without detection. This paper proposes TraceGuard, a verification framework that pairs a per-unit QR code with an immutable blockchain ledger so that authenticity checks do not depend on trusting the manufacturer's own database. Each product is assigned a hashed identifier at the point of manufacture; the hash, together with batch and ownership metadata, is written to a permissioned blockchain through a smart contract, and the QR code printed on the product encodes only a lookup reference, never the underlying data. A scanning application resolves the reference against the ledger and returns a verification result along with the product's custody history, so a buyer can see not just whether a code is genuine but whether it has already been scanned as sold elsewhere. We describe the smart contract design, the QR encoding scheme, the mobile verification workflow, and a batch-transfer mechanism for supply-chain custody changes, and we evaluate the system on constructed product batches for verification latency, duplicate-scan detection, and resistance to QR cloning. The results indicate that ledger-backed verification catches duplicate and out-of-sequence scans that a static-database approach would miss, at a modest latency cost.

**Index Terms** - Counterfeit Detection, QR Code Verification, Blockchain Traceability, Smart Contracts, Supply Chain Transparency, Product Authentication

## I. INTRODUCTION

### 1.1 Background and Motivation

Counterfeit goods span pharmaceuticals, electronics, apparel, and packaged consumer products, and the

problem is not confined to any one price tier. A buyer has almost no independent way to tell a genuine item from a convincing copy at the point of sale, because the manufacturer's own claim of authenticity is usually the only signal on offer [1]. Governments and industry

bodies have documented the scale of the issue for years, but the tooling available to an ordinary buyer has not moved much beyond a printed serial number or a hologram sticker, both of which can be reproduced once a counterfeiter has physical access to a genuine unit [2].

Two technologies have been proposed separately to close this gap. QR codes give a cheap, high-density way to attach a per-unit identifier to a product, and blockchain ledgers give a way to record that identifier so that no single party, including the manufacturer, can quietly edit the record after the fact [3]. Used together, a QR code becomes a pointer into a ledger rather than a self-contained claim, which changes what a counterfeiter has to defeat.

## 1.2 Problem Statement

We set out to build a verification system with three properties. First, the authenticity record for a product must live somewhere a single actor, including the brand that issued it, cannot unilaterally rewrite. Second, a scan must reveal not just whether a code is well-formed but whether that specific unit has already been claimed as sold, so that a cloned code printed on a second, fake unit is caught even though the code itself decodes correctly. Third, the system must work with an ordinary smartphone camera on the buyer's side, with no special hardware and no app permissions beyond camera access.

## 1.3 Contributions

The contributions of this paper are as follows. We present a product-registration scheme in which a manufacturer commits a cryptographic hash of each unit's identifying data to a smart contract, rather than the data itself, so the ledger stores a verifiable fingerprint without exposing manufacturing details. We describe a QR encoding scheme in which the printed code carries only an opaque lookup reference, so cloning the printed pattern does not by itself grant the ability to mark a unit as sold. We define a custody-transfer mechanism that lets distributors and retailers record a change of possession on the same ledger, which produces an auditable chain of custody rather than a single manufacture-time stamp. Finally, we

evaluate the system on constructed product batches, measuring verification latency, duplicate-scan detection, and behaviour under a simulated cloning attempt.

## 1.4 Organisation of the Paper

Section II reviews centralized authentication systems, blockchain traceability work, and known QR code limitations. Section III sets out the system architecture. Section IV details the methodology: registration, smart contract design, QR encoding, verification, and custody transfer. Section V covers the implementation. Section VI presents the evaluation. Section VII discusses the trust argument behind the design, Section VIII states the limitations, and Section IX concludes with future work.

## II. RELATED WORK

### 2.1 Centralized and Database-Backed Authentication

The dominant anti-counterfeiting approach today ties a serial number or hologram to a manufacturer's own database, queried either directly or through an SMS/web lookup [1]. This works only as long as the buyer trusts that the manufacturer's records have not been altered, and it gives no independent way to detect a case where a genuine serial number has simply been reprinted onto multiple counterfeit units, since the database has no notion of how many times a given code has already been presented as sold [2].

### 2.2 Blockchain-Based Traceability Systems

Blockchain traceability has been explored extensively for food and pharmaceutical supply chains, where the goal is usually to record provenance events, such as harvest, processing, and shipment, on a shared ledger that no single participant controls [3], [4]. Smart-contract platforms such as Ethereum and permissioned frameworks such as Hyperledger Fabric are both used in this literature, generally chosen on the basis of whether the deployment needs public verifiability or restricted participation [5], [6]. Most of this prior work records batch-level events rather than a per-unit

consumer-facing verification step, which leaves the last mile of the supply chain, the point where an individual buyer checks a single item, largely unaddressed.

### 2.3 QR Code Security Limitations

A QR code is a public, well-documented encoding standard [7], and anything printed on a product can be photographed and reprinted by anyone with a decent camera and printer. Static QR schemes that embed a URL or a plain serial number inherit this weakness directly: once a genuine code is copied, every copy verifies identically, because verification checks only whether the code is well-formed, not how many times it has already been used [8]. Some commercial schemes mitigate this with tamper-evident labels that are destroyed on removal, but that is a physical control layered on top of the code, not a property of the code itself.

### 2.4 Gap Addressed

The gap we address sits between these two bodies of work. We adopt the ledger-based integrity guarantees explored in blockchain traceability research, but apply them at the granularity of a single consumer-facing scan rather than a batch event, and we pair that with a QR encoding scheme designed so that copying the printed pattern does not automatically defeat the duplicate-scan check that the ledger makes possible.

## III. SYSTEM ARCHITECTURE

### 3.1 Overall Structure

TraceGuard has three participants: the manufacturer, who registers products; the buyer or downstream custodian, who scans a code to verify or take custody of a unit; and the ledger itself, which neither party can unilaterally rewrite. The system is split into a registration path, used once per unit at the point of manufacture, and a verification/custody path, used every time the unit changes hands or is checked by a buyer.

### 3.2 Component Overview

The manufacturer-side registration service generates a unique product identifier, computes its hash together

with batch metadata, and submits a registration transaction to the smart contract. The smart contract stores the hash, an initial owner address, and a status flag, and exposes functions to read status, transfer custody, and mark a unit as sold. The QR generation module produces a printable code encoding an opaque reference token, not the hash itself and not any manufacturing data. The mobile verification application scans a code, resolves the reference to a contract query, and displays the result together with the custody history retrieved from the ledger's event log.

### 3.3 Data Flow

At manufacture time, the registration service writes a record to the contract and encodes the returned reference into a QR code, which is printed on the unit. At each custody change, the current holder signs a transfer transaction identifying the new holder, which the contract accepts only if the caller is the recorded current holder. At verification time, the buyer's scan reads the reference, queries the contract for status and history, and the application renders a plain verified/not verified/already sold result along with the ownership chain, without ever exposing the underlying registration data to the buyer.

## IV. METHODOLOGY

### 4.1 Product Registration and Hash Generation

For each unit, the manufacturer generates a random 128-bit identifier and combines it with batch number, product SKU, and manufacture timestamp into a single record. A SHA-256 hash of this record is computed locally and submitted to the smart contract; the plaintext record itself is kept in the manufacturer's own system and is never transmitted to the ledger. This means the chain stores a fingerprint that can confirm a later-presented record matches what was registered, without the chain itself becoming a public catalogue of batch data.

### 4.2 Smart Contract Design

The contract exposes four functions: `registerProduct(hash, initialOwner)`, `transferCustody(productId, newOwner)`,

markSold(productId), and getHistory(productId). Each state-changing call is restricted to the address currently recorded as the owner, enforced with a contract-level modifier rather than left to the calling application, so a compromised front end cannot itself alter custody. Every call emits an event, and the concatenation of events for a given product ID is what getHistory returns, giving an append-only audit trail rather than a single mutable status field.

#### 4.3 QR Code Encoding Scheme

The printed QR code encodes only a short opaque reference token, generated independently of the product hash, that the verification application maps to a product ID through a lookup rather than by decoding the token directly. This separation means that photographing and reprinting a QR code copies the token but does not grant the counterfeiter any way to alter the ledger's custody or sold status for the corresponding product ID; a cloned code still resolves to the same underlying record, so if that record has already been marked sold, every clone reports the same already-sold result.

#### 4.4 Verification Workflow

When a buyer scans a code, the application reads the reference token, queries the contract for the product's current status, and displays one of three outcomes: verified and available, verified and previously marked sold under a different transaction, or reference not found. The second outcome is the one a static database cannot reliably produce, because it requires the system to remember every prior verification-as-purchase event rather than just the existence of a valid code.

#### 4.5 Custody Transfer and Ownership Tracking

Distributors and retailers use the same application, authenticated with a wallet address issued to their organisation, to call transferCustody when they receive a shipment and markSold when a unit is purchased by an end consumer. Because each transfer requires a signature from the current recorded owner, custody cannot be reassigned by a party who has merely obtained a copy of the QR code, only by the party the ledger currently recognises as holding the unit.

#### 4.6 Evaluation Method

We constructed test batches of simulated products and ran three experiments: measuring end-to-end verification latency from scan to displayed result; simulating duplicate scans of the same code to confirm the second scan is reported as already sold; and simulating a cloning attempt in which a second physical unit is printed with a copied QR code, to confirm that marking the original as sold is reflected when the clone is scanned.

### V. IMPLEMENTATION

#### 5.1 Technology Stack

The smart contract is written in Solidity and deployed to a permissioned Ethereum-compatible test network. The registration service and API layer are built in Node.js with Express, communicating with the contract through the web3.js library. The mobile verification application is a React Native client with a built-in QR scanner. Off-chain metadata that does not need to be on the ledger, such as product images and marketing copy, is stored in a conventional relational database and linked to the on-chain record by product ID.

#### 5.2 Smart Contract Deployment

The contract was deployed with a fixed gas limit per call and tested against the four core functions under a simulated load of concurrent registrations. Access control is enforced with Solidity modifiers checking the caller's address against the recorded owner for every state-changing function, and all functions revert rather than silently fail when a precondition is not met.

#### 5.3 Mobile Application

The scanning application decodes the QR token, resolves it through the API layer to a product ID, and issues a read-only contract call to fetch status and history. No verification step requires the buyer to hold a wallet or pay a transaction fee, since read calls do not modify chain state; only manufacturer, distributor, and retailer actions, which change custody, require a signed transaction.

#### 5.4 Data Handling and Security

Private keys for manufacturer and distributor accounts are held in a hardware-backed keystore rather than in application code, and every state-changing API request is authenticated separately from the on-chain signature check, so a compromised API credential alone cannot forge a valid transfer. The reference token embedded in the QR code is generated with a cryptographically secure random source and is long enough that guessing a second valid token is not a practical attack.

## VI. RESULTS AND EVALUATION

### 6.1 Verification Latency

Across repeated scans on the test network, the median time from scan to displayed verification result was under two seconds, dominated by the read-only contract call and network round trip rather than by QR decoding, which completes in a fraction of that time on a modern smartphone camera.

### 6.2 Duplicate-Scan Detection

In constructed batches where the same code was scanned a second time after a markSold call, the application correctly reported the second scan as already sold in every trial, which a lookup against a static, non-eventful database would not do unless it separately tracked a purchase flag that the same database administrator could also edit.

### 6.3 Resistance to QR Cloning

When a second physical unit was printed with a copied reference token to simulate a cloning attempt, scanning the clone after the original had been marked sold returned the already-sold result rather than a fresh verified result, because both codes resolve to the same underlying ledger record. This confirms that copying the printed pattern alone does not let a counterfeiter present a second unit as new and unsold.

### 6.4 Comparison with Centralized Approach

We compared TraceGuard against a simulated static-database baseline that stores a serial number and a boolean sold flag. The baseline correctly rejected malformed codes but could not distinguish a first legitimate sale from a subsequent presentation of a

cloned code once the flag had been reset or was never set, since the flag by itself carries no history and can be altered by anyone with write access to the database. The ledger-backed approach preserves the full event history, which is what makes the already-sold determination reliable rather than dependent on a single mutable field.

## VII. DISCUSSION

The central design choice in TraceGuard is treating the QR code as a pointer rather than a claim. A static code that encodes a serial number is only as trustworthy as the database behind it, and that database is controlled by the same party whose products are being verified, which is an inherent conflict of interest whenever a manufacturer might be tempted to under-report counterfeits. Moving the authoritative record to a ledger that the manufacturer can write to but not unilaterally rewrite does not eliminate every incentive problem, but it does mean that a change to a product's recorded status leaves a permanent, independently auditable trace, rather than a silent edit.

A second point worth making explicitly is that the system does not claim to stop physical counterfeiting outright. A well-resourced counterfeiter can still manufacture a convincing physical copy of a product; what the ledger changes is whether that copy can also be presented as a genuine, unsold unit. By separating the printed code from the authenticity claim, the design shifts the attacker's problem from copying a pattern to defeating a cryptographic and access-controlled ledger, which is a materially harder problem.

## VIII. LIMITATIONS

This work has four limitations we want to state plainly. The evaluation was run on constructed test batches on a permissioned test network, not on a production deployment with real manufacturers and real buyers, so latency and throughput figures may not hold under production load or on a public chain with variable gas prices. The system assumes the manufacturer's registration step is honest; if a manufacturer registers a

hash for a counterfeit unit in the first place, the ledger will faithfully verify that unit, because the ledger can only attest to what was written to it, not to the physical provenance of the item described. The custody-transfer mechanism assumes distributors and retailers hold and protect their private keys correctly, and we have not evaluated the system's behaviour under key loss or compromise at scale. Finally, we have not run a field trial with an adversarial party attempting to defeat the system outside of the cloning simulation described in Section VI.

## IX. CONCLUSION AND FUTURE WORK

### 9.1 Conclusion

We presented TraceGuard, a counterfeit-detection framework that pairs a per-unit QR code carrying only an opaque reference with a blockchain ledger that records registration, custody transfer, and sale status as an append-only event history. On constructed test batches, the system reliably distinguished a first genuine scan from a duplicate or cloned presentation of the same underlying record, a distinction a static-database approach cannot make without itself relying on a single mutable flag. The core contribution is architectural rather than cryptographic: separating what is printed on the product from what is authoritative about it.

### 9.2 Future Work

Future work includes a field pilot with a manufacturer partner to measure performance under real transaction volume and gas costs on a public or consortium chain, an evaluation of key-management schemes for distributors and retailers that reduces the operational burden of protecting private keys, and an extension of the custody model to handle returns and refurbishment, where a unit legitimately re-enters the supply chain after having already been marked sold.

## REFERENCES

- [1] OECD/EUIPO, "Trends in Trade in Counterfeit and Pirated Goods," Illicit Trade, OECD Publishing, Paris, 2019.
- [2] World Health Organization, "A Study on the Public Health and Socioeconomic Impact of Substandard and Falsified Medical Products," WHO, Geneva, 2017.
- [3] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," IEEE Access, vol. 4, pp. 2292-2303, 2016.
- [4] F. Tian, "An Agri-Food Supply Chain Traceability System for China Based on RFID and Blockchain Technology," in Proc. 13th Int. Conf. Service Systems and Service Management (ICSSSM), 2016.
- [5] V. Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," Ethereum White Paper, 2014.
- [6] Hyperledger Foundation, "Hyperledger Fabric: A Blockchain Platform for the Enterprise," Linux Foundation Project Documentation, 2020.
- [7] ISO/IEC 18004:2015, "Information Technology - Automatic Identification and Data Capture Techniques - QR Code Bar Code Symbol Specification," International Organization for Standardization, 2015.
- [8] I. Tkachenko, W. Puech, C. Destruel, O. Strauss, J.-M. Gaudin, and C. Guichard, "Two-Level QR Code for Private Message Sharing and Document Authentication," IEEE Transactions on Information Forensics and Security, vol. 11, no. 3, pp. 571-583, 2016.
- [9] N. Szabo, "Formalizing and Securing Relationships on Public Networks," First Monday, vol. 2, no. 9, 1997.