Blockchain-Based Supply Chain Authentication for Luxury and Collectible Goods

Hsuan-Chi Chang, Wei-Ju Li
Virginia Tech
Alexandria, VA, USA
hsuanchi@vt.edu,weijuli@vt.edu

ABSTRACT

The global counterfeit market is estimated to exceed $1.7 trillion annually. [1], which means that buying a luxury bag from the second-hand market carries a high risk of purchasing a counterfeit product. Traditional paper certification, laser labels, NFC chips inside the bag, and even packaging boxes can all be forged. In general, customers lack the capability to authenticate the authenticity of these products.

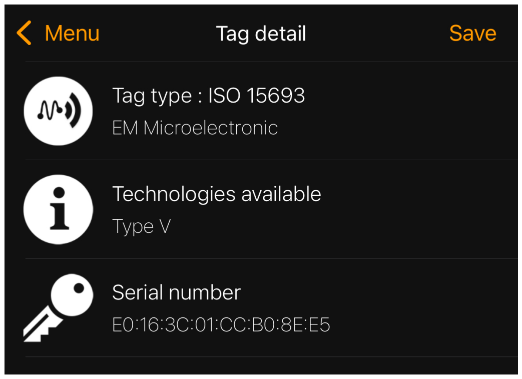
Therefore, we want to find an innovative solution to address this problem. By storing production and sales history data on Polygon (a Layer 2 scaling solution for Ethereum that provides lower transaction fees, higher throughput, and full compatibility with Ethereum smart contracts), all customers can access this information by scanning the NFC tag inside their bags.

Moreover, to prevent counterfeit NFC tags, we have decided to implement Ed25519 digital signature technology (a widely used, highly secure elliptic-curve cryptographic algorithm known for its efficiency and strong security guarantees). After scanning the NFC tag, the system will go through a digital signature verification process (a method where a cryptographic public-private key pair ensures that the tag was issued by a legitimate source), allowing customers to verify the authenticity of their purchase with confidence.

ACM Reference Format:
Hsuan-Chi Chang, Wei-Ju Li. 2025. Blockchain-Based Supply Chain Authentication for Luxury and Collectible Goods. In . ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The global counterfeit market is estimated to exceed $1.7 trillion annually. [1] Currently, verifying the authenticity of luxury goods relies on human experts or trusting the seller. Luxury brands like LVMH and Chanel use NFC tags inside their bags and purses. However, these NFC tags can be easily counterfeited, and while customers can scan them to get information (such as the serial number) 1, they cannot verify authenticity by themselves using this number. Therefore, we aim to enhance NFC security using digital signatures. Additionally, we want to store production and sales history data on the Polygon to track the product's history. Even if the bag changes hands multiple times, its authenticity can still be traced and verified, ensuring the records remain tamper-proof.



Tag type : ISO 15693
EM Microelectronic
Technologies available
Type V
Serial number
E0:16:3C:01:CC:B0:8E:E5

Figure 1: Screenshot after scanning a genuine LV coin purse, showing an unverifiable Serial Number.

2 APPROACH OVERVIEW

Our solution brings together blockchain technology, cryptographic validation, and decentralized applications (dApps) to provide a secure and scalable luxury product authentication system.

For blockchain infrastructure, we use Polygon, which is very scalable and has minimal transaction fees, ideal for mass adoption. Instead of storing all product information on-chain, we only store required metadata and decentralized storage pointers, maintaining blockchain storage efficiency while tamper-proof provenance tracking is ensured. This minimizes on-chain storage cost significantly while ensuring that each product's authentication record remains verifiable and immutable, both data integrity and consumer trust intact.

In terms of cryptographic authentication, we apply Ed25519[3][4] as the scheme of digital signature, selected on the basis of efficiency, resistance to side-channels, and appropriateness for implementation alongside low-power NFC chips. Through this, it becomes possible for the NFC-enabling product authentication to be prompt, secure, and resistant to forgery, which allows practical deployment in real-world luxury object verification.

Considering user interactions, we separate the businesses' and consumers' part for enhanced security and misuse elimination. Shippers can scan merchandise with a dApp to find cryptographically secured records of provenance and trade history on a blockchain. Corporations, however, will sign up their products through another, controlled system rather than the identical dApp. This distinction

Conference'17, July 2017, Washington, DC, USA
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ensures that only proper corporations are capable of adding goods on-chain while avoiding the uploading threat of phantom data, supporting trust in the system.

For scalability and adoption, Polygon's Layer 2 solutions allow us to handle massive numbers of transactions with no cost or speed compromise. To supplement the utilization of Polygon's scalability capabilities, we also have a business platform service, which simplifies the process of registering products and makes it accessible. This allows brands to onboard their products onto the blockchain with ease without requiring sophisticated technical expertise. On the user end, our user-friendly dApp interface lowers the threshold of product legitimacy, providing a hassle-free and effortless experience.

By incorporating scalability, decentralization storage optimization, cryptographic security, and user accessibility, our platform provides a practical and real-world solution for combating counterfeiting within the luxury goods sector.

3 DESIGN

Our system follows a modular three-layer design that separates user-facing applications, core logic, and decentralized infrastructure. Figure 2 illustrates the architecture and end-to-end workflow of our blockchain-based authentication system.

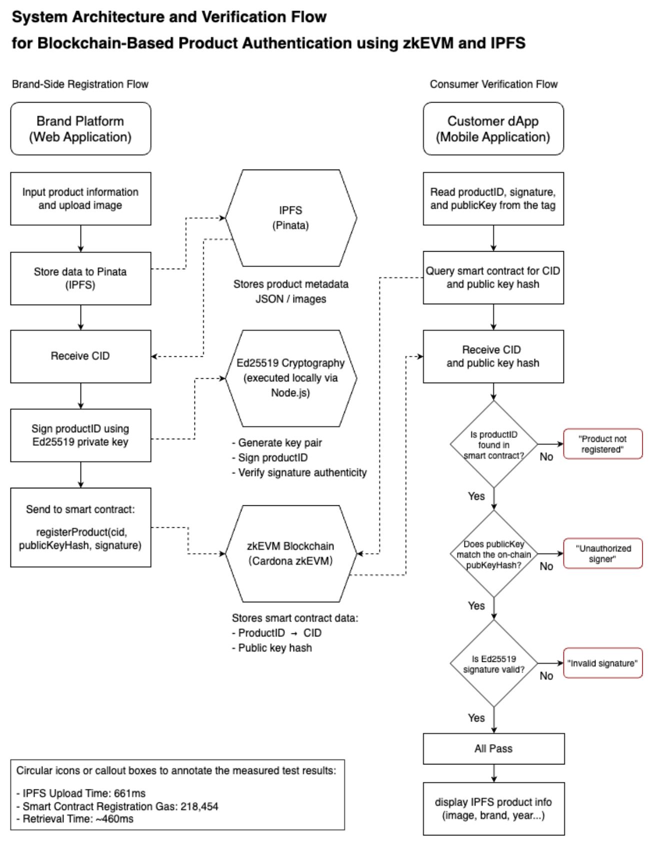

System Architecture and Verification Flow
for Blockchain-Based Product Authentication using zkEVM and IPFS
Brand-Side Registration Flow
Consumer Verification Flow
Brand Platform (Web Application)
Input product information and upload image
Store data to Pinata (IPFS)
Receive CID
Sign productID using Ed25519 private key
Send to smart contract: registerProduct(cid, publicKeyHash, signature)
IPFS (Pinata)
Stores product metadata JSON / images
Ed25519 Cryptography (executed locally via Node.js)
- Generate key pair
- Sign productID
- Verify signature authenticity
zkEVM Blockchain (Cardona zkEVM)
Stores smart contract data:
- ProductID → CID
- Public key hash
Customer dApp (Mobile Application)
Read productID, signature, and publicKey from the tag
Query smart contract for CID and public key hash
Receive CID and public key hash
Is productID found in smart contract?
No
"Product not registered"
Yes
Does publicKey match the on-chain pubKeyHash?
No
"Unauthorized signer"
Yes
Is Ed25519 signature valid?
No
"Invalid signature"
Yes
All Pass
display IPFS product info (image, brand, year...)
Circular icons or callout boxes to annotate the measured test results:
- IPFS Upload Time: 661ms
- Smart Contract Registration Gas: 218,454
- Retrieval Time: ~460ms

Figure 2: System Architecture and Verification Flow

3.1 Architecture Overview

The system consists of three main layers:

- **Client Interaction Layer:** Provides interfaces for both brand users and consumers. This includes:
  - A web platform for brands to upload product data and initiate registration.
  - A mobile dApp for consumers to scan NFC tags and verify authenticity.
- **Logic and Processing Layer:** Executes the core application logic:
  - IPFS API calls for metadata storage and CID retrieval.
  - Ed25519 signing and signature verification (executed locally via Node.js or browser-side JS).
  - Interaction with the smart contract's ‘registerProduct‘ and ‘getProduct‘ functions.
- **Infrastructure Layer:** Provides decentralized and verifiable foundations:
  - Polygon zkEVM blockchain for immutable smart contract storage and product records.
  - IPFS (via Pinata) for decentralized metadata hosting.
  - Ed25519 cryptography libraries (TweetNaCl) for client-side and server-side cryptographic operations.

3.2 Workflow Description

The full system flow proceeds as follows:

(1) The brand inputs product metadata (e.g., model, material, production year) through a web platform and uploads it to IPFS.
(2) The IPFS service returns a content identifier (CID), which uniquely maps to the stored product metadata.
(3) The brand then signs the productID using its private Ed25519 key and sends the signature, public key hash, and CID to the smart contract via the `registerProduct` function.
(4) The smart contract stores the CID and the hash of the public key on-chain, binding the productID to verifiable data.
(5) The product is physically shipped with an embedded NFC tag containing the signed productID, the raw public key, and the digital signature.
(6) A consumer scans the NFC tag using the dApp, retrieves the productID, public key, and signature, and queries the smart contract for the expected public key hash and CID.
(7) The dApp performs client-side Ed25519 signature verification to validate the authenticity of the tag.
(8) If the verification passes, the dApp fetches product metadata from IPFS using the CID and displays it to the user.

3.3 Design Principles

The architecture is designed to prioritize:

- **Decentralization:** Consumers independently verify product authenticity without relying on brand servers.
- **Efficiency:** Only critical metadata is stored on-chain, while large files (e.g., product images) are offloaded to IPFS.
- **Security:** Forged tags are reliably detected through Ed25519 signature verification and strict smart contract control.

- **Scalability:** The use of Polygon zkEVM ensures high throughput and low transaction cost for brand-side operations.

This layered architecture supports tamper-resistant verification while remaining cost-effective and user-friendly for both brands and consumers.

3.4 Security Considerations

The design of our authentication system emphasizes security at both the cryptographic and architectural levels. First, all product records are signed using Ed25519 digital signatures, which are known for their performance, small key sizes, and strong resistance to side-channel attacks. This cryptographic layer ensures that only a legitimate brand possessing the private key can produce valid product identifiers.

To prevent unauthorized product registration, the smart contract employs access control via the OpenZeppelin Ownable module. Only the contract owner can register brands and products, eliminating the risk of arbitrary submissions from external addresses. Furthermore, the productID is enforced as unique, and product entries are immutable after registration.

At the data level, we use on-chain storage to bind each productID to its CID and the brand's public key hash. During verification, the system checks both the public key hash and the signature to detect forgery attempts. Unlike QR-code-based solutions, which can be easily duplicated, our design binds identity to a cryptographically verifiable token.

These design decisions help protect against common threats such as replay attacks, unauthorized product cloning, or signature spoofing.

3.5 User Interface Wireframe Design

To demonstrate the practical implementation of both brand and consumer workflows, we designed a simple front-end interfaces for users. The consumer-facing dApp enables NFC scanning.

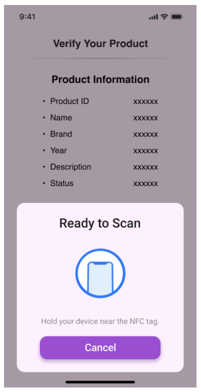


Figure 3: Consumer-side dApp interface scanning NFC tag

4 IMPLEMENTATION

To demonstrate the feasibility of our authentication framework, we implemented a working prototype using the Cardona zkEVM testnet, IPFS decentralized storage (via Pinata), and Ed25519 digital signature verification. The implementation primarily focuses on demonstrating data registration, signature generation, smart contract execution, and consumer-side verification. The backend server is implemented using Node.js and exposes RESTful APIs for brand/product registration, signature generation, and verification. A code excerpt is included in the Appendix.

4.1 Smart Contract Architecture

The smart contract was developed in Solidity and deployed to the Cardona zkEVM testnet. It includes functions for brand registration and product registration. The contract stores a mapping from `productID` to a struct that includes the CID of metadata stored on IPFS and the hash of the public key used for signing. Key functions include:

- `registerBrand(address brand, bytes32 pubKeyHash):` Registers a brand and its associated Ed25519 public key hash.
- `registerProduct(string productID, string cid):` Links a productID to its IPFS CID and stores it in the blockchain.
- `getProduct(string productID):` Allows consumers to retrieve the stored CID and public key hash for verification.

To ensure only authorized brands can register products, we implemented modifier-based access control, and brand-public key associations are immutable after registration.

4.2 IPFS Integration

Product metadata—including productID, brand name, model, material, year, and image hash—is stored as JSON and uploaded to IPFS using the Pinata SDK. Upon upload, a content identifier (CID) is returned and stored in the smart contract. IPFS allows data to be stored off-chain in a decentralized, tamper-resistant manner, while the on-chain CID acts as a verifiable reference.

4.3 Digital Signature and Tag Simulation

We used the `tweetnacl` cryptographic library in Node.js to simulate Ed25519-based digital signatures. A key pair is generated for each brand, and each `productID` is signed using the brand's private key. The output signature, public key, and productID are then Base64-encoded and stored in a JSON object that simulates the content of an NFC tag.

For the verification process, the consumer-facing dApp reads the mock JSON object, extracts the `productID`, `signature`, and `publicKey`, then queries the smart contract for the corresponding CID and public key hash. It reconstructs the original message and verifies the Ed25519 signature locally. This simulation not only models the content embedded in NFC tags but also demonstrates the feasibility of performing complete verification using only local resources and blockchain data.

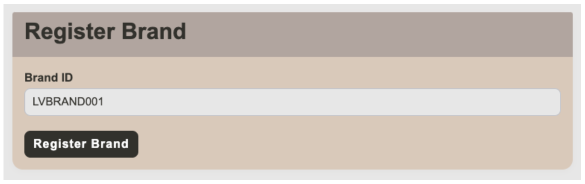
4.4 Forgery Simulation and Error Handling

To test the system's robustness, we introduced forged entries by altering the productID in 10% of the simulated tag data. Signature verification failed for all modified entries, demonstrating that forged tags can be reliably rejected using local cryptographic validation without relying on centralized servers or external authorities.

4.5 Prototype Interface and User Flow

To demonstrate the practical use of the system, we developed a three-step user flow that simulates both the brand-side and consumer-side interactions. The prototype supports the full cycle from brand onboarding to product registration and consumer-side verification.

4.5.1 Brand Creation. Brand owners begin by creating their identity in the system. Through the brand platform interface, a brand can register its name and generate a public-private Ed25519 key pair. The public key hash is sent to the smart contract using the `registerBrand` function. The interface confirms successful registration by displaying the transaction hash.



Register Brand

Brand ID: LVBRAND001

Figure 4: Brand creation interface

4.5.2 Product Registration. After registering the brand, the owner can upload a product by filling out a metadata form including product ID, model name, year, and material. Upon submission, the data is serialized into JSON and uploaded to IPFS via the Pinata API. The CID returned from IPFS is displayed in the UI, and the productID is signed locally using the brand's private key. The `registerProduct` function is then called to store the CID and public key hash on-chain.

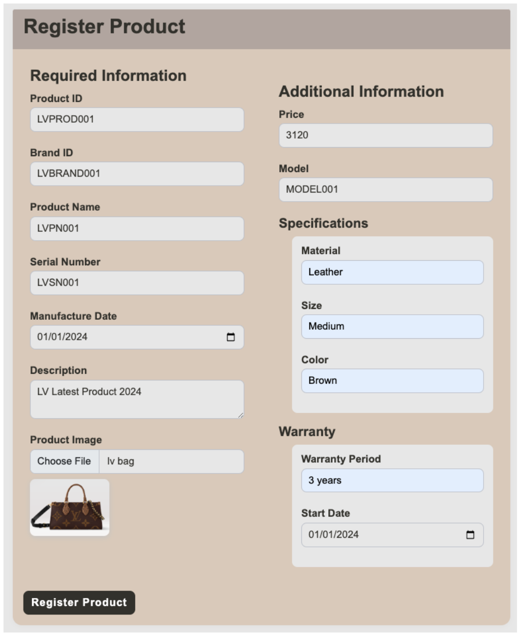
4.5.3 Product Verification. On the consumer side, the user interface allows a customer to manually input tag data into the verification dApp. Instead of reading from a physical NFC tag or a local JSON file, the interface presents a form where the user enters the following fields extracted from the tag:

- **Product ID**
- **Brand ID**
- **Serial Number**
- **Signature** (Base64-encoded)
- **Tag Public Key**

After submitting the form, the frontend sends the input data to a backend verification API. The server performs the following verification steps:

(1) Retrieves the expected CID and registered public key hash for the given Product ID and Brand ID from the smart contract.
(2) Compares the submitted public key against the on-chain hash to confirm signer authenticity.
(3) Uses the Ed25519 signature verification algorithm to validate the signature against the product ID and serial number.

If all checks pass, the system returns a success message along with the product's metadata retrieved from IPFS. Otherwise, the UI displays a descriptive error message, such as "Invalid Signature" or "Unauthorized Signer," based on the verification failure point.



Register Product

Required Information

Product ID: LVPROD001
Brand ID: LVBRAND001
Product Name: LVPN001
Serial Number: LVSN001
Manufacture Date: 01/01/2024
Description: LV Latest Product 2024
Product Image: Choose File | lv bag

Additional Information

Price: 3120
Model: MODEL001

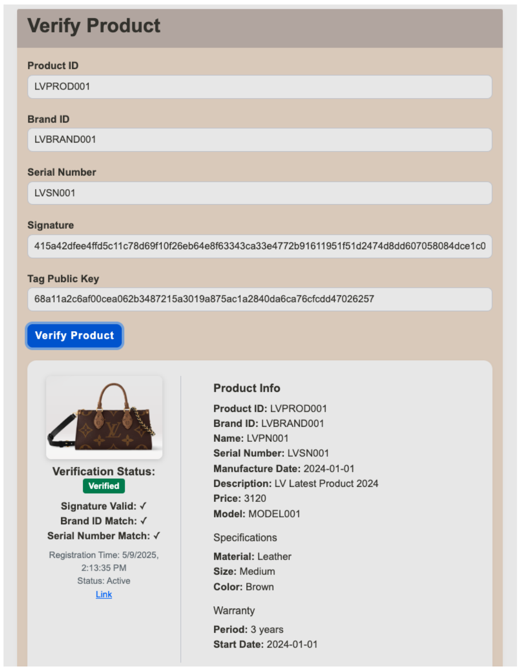
Specifications

Material: Leather
Size: Medium
Color: Brown

Warranty

Warranty Period: 3 years
Start Date: 01/01/2024

Figure 5: Product metadata input and registration interface



Verify Product

Product ID: LVPROD001
Brand ID: LVBRAND001
Serial Number: LVSN001
Signature: 415a42dfee4ffd5c11c78d69f10f26eb64e8f63343ca33e4772b91611951f51d2474d8dd607058084dce1c0
Tag Public Key: 68a11a2c6af00cea062b3487215a3019a875ac1a2840da6ca76cfcdd47026257

Verification Status: Verified
Signature Valid: ✓
Brand ID Match: ✓
Serial Number Match: ✓
Registration Time: 5/9/2025, 2:13:35 PM
Status: Active
Link

Product Info
Product ID: LVPROD001
Brand ID: LVBRAND001
Name: LVPN001
Serial Number: LVSN001
Manufacture Date: 2024-01-01
Description: LV Latest Product 2024
Price: 3120
Model: MODEL001

Specifications
Material: Leather
Size: Medium
Color: Brown

Warranty
Period: 3 years
Start Date: 2024-01-01

Figure 6: Consumer-side verification interface with form input and result display

5 EVALUATION

5.1 Test Environment

We conducted a series of performance evaluations to validate the feasibility and efficiency of the proposed system. All experiments were executed on the Cardona zkEVM test network, and IPFS data was hosted via the Pinata. The smart contract was deployed to address `0xe391aDF6Df8075D198C37b7B0cAC8C82fc4cE2BC`. To simulate realistic usage, three different data payloads were prepared: small (200 bytes), medium (371 bytes), and large (2261 bytes).

5.2 IPFS Storage Performance

We measured the upload time and actual IPFS storage size for each data sample. Table 1 summarizes the results.

Table 1: IPFS Storage Performance

Size	Time (ms)	Stored (B)	Efficiency
Small (200B)	661	259	1.30
Medium (371B)	391	430	1.16
Large (2261B)	436	2320	1.03

Analysis: As data size increases, the storage efficiency improves. Upload times are relatively stable and not significantly affected by payload size. The final storage size is linearly proportional to the original data size.

5.3 Smart Contract Performance

We measured gas usage and transaction latency for both brand registration and product registration operations. Table 2 shows the findings.

Table 2: Smart Contract Gas and Latency

Size	Latency (ms)	Brand Gas	Prod. Gas	Total
Small	12875	50283	168171	218454
Medium	18190	50283	168171	218454
Large	18331	50283	168171	218454

Analysis: Gas consumption remains constant regardless of data size, as the smart contract only stores the CID. Latency increases slightly with larger payloads but remains within an acceptable range. Brand registration always costs 50,283 gas, while product registration costs 168,171 gas.

5.4 Data Retrieval Performance

We simulated product lookups using the product ID and evaluated retrieval time and success rate. Table 3 summarizes the outcomes.

Table 3: Data Retrieval Performance

Data Size	Retrieval Time (ms)	Success Rate
Small	463	100%
Medium	427	100%
Large	475	100%

Analysis: Retrieval time was consistent across all data sizes, ranging from 400–500ms. All records were successfully retrieved, validating the reliability of the smart contract and decentralized storage integration.

5.5 Evaluation Strategy and Assumptions

All experiments were conducted on the Cardona zkEVM testnet using a local Node.js backend and browser-based dApp frontend. We used the Pinata IPFS gateway for metadata uploads and retrieval, and the tweetnacl library for all Ed25519 operations.

We selected three dataset sizes—small (200B), medium (371B), and large (2261B)—to represent realistic product metadata ranges. These correspond to simple text-based products, medium records with branding and collection information, and full records including image hashes and descriptions.

We also simulated tampered entries in 10% of verification attempts to test signature resilience. All 10 forged signatures were correctly rejected. Although we did not integrate hardware-based NFC, the form-based simulation reflects the logical structure of actual tag scanning. Future testing will include physical tags and real-time latency profiling across devices.

5.6 Signature Verification Performance

To evaluate the feasibility and efficiency of Ed25519 digital signatures , we conducted an experiment simulating NFC tag verification. In this phase, we did not use physical NFC hardware. Instead, we emulated NFC tag contents using locally stored JSON files. Since cryptographic NFC tags (10 tags cost $50 and beyond) are more expensive than conventional NFC tags and not readily available for prototyping purposes.

Each simulated tag contains JSON-formatted product metadata, including a serial number, product ID, and timestamp. The data is signed using the Ed25519 algorithm via the `tweetnacl` library in Node.js. The resulting signature and public key are Base64-encoded and stored along with the data in a single JSON file representing the emulated tag.

For each test iteration, the system reads the mock NFC JSON file, reconstructs the original message as a UTF-8 encoded byte array, and performs signature verification using the detached verification function. To simulate real-world tampering, every 10th message is deliberately modified by altering a byte in the serialized data prior to verification.

Algorithm 1 Ed25519 Tag Verification

```
1: Generate keypair $(pk, sk)$ using Ed25519
2: Serialize message $m$ to string
3: Convert string to byte array $b$
4: Sign $b$ using $sk$ to get signature $s$
5: Store $\{m, s, pk\}$ as JSON
6: for $i = 1$ to 100 do
7:     if $i \bmod 10 = 0$ then
8:         Modify $m$ to simulate forgery
9:     end if
10:    Recompute $b$ from $m$
11:    Verify $s$ with $b$ and $pk$
12:    Record result and timing
13: end for
```

Two key metrics were recorded during testing:

- **Signature verification success rate**

- **Average verification latency (ms)**

The experiment yielded the following results:

- Successful verifications: **90**
- Failed verifications: **10**
- Verification success rate: **90.00%**
- Average verification time: **5.53 ms**

These results indicate that the system effectively detects forged or tampered messages, with all 10 deliberately altered entries correctly rejected.

5.7 Cost Analysis

Gas Costs:

- Brand registration: 50,283 gas
- Product registration: 168,171 gas
- Total per product: 218,454 gas

Storage Costs:

- IPFS: Free (using Pinata's free tier)
- On-chain: Only CID stored, fixed size

5.8 Summary

Our evaluation validates the system's performance across storage, cost, verification accuracy, and latency. These findings suggest that the system can be securely and efficiently deployed for real-world usage, with reliable performance across blockchain operations and local signature verification.

6 DISCUSSION

Although our system demonstrates technical feasibility and effective performance, several practical considerations remain. First, the current prototype does not include real NFC tag hardware due to cost constraints. Integrating physical tags and ensuring cross-device compatibility will be addressed in future iterations.

Second, while the use of public key hashes prevents unauthorized registration, key revocation and rotation remain open problems. A brand may need to rotate keys periodically, which requires contract-level support for updating trusted hashes without compromising security.

Additionally, although IPFS provides decentralized storage, availability and latency may fluctuate without persistent pinning. A possible solution involves using IPFS pinning services with SLA guarantees or shifting toward hybrid storage (e.g., Filecoin or Arweave integration).

Finally, user adoption is a critical success factor. Future work will include usability testing and security audits to ensure our dApp is robust, intuitive, and secure under real-world usage scenarios.

7 RELATED WORK

7.1 Existing Solutions

Several existing approaches have been developed to improve product authentication and supply chain transparency, including both industry-backed solutions and academic research.

7.1.1 Aura Consortium Blockchain and LV Diamonds Certificate. Aura Consortium Blockchain is a private, permissioned blockchain developed by luxury brands such as LVMH, Prada, and Cartier. It is designed to provide product authenticity, traceability, and responsible sourcing verification. Unlike public blockchains, Aura operates as a closed network where only approved entities can validate transactions and store product metadata.

The LV Diamonds Certificate [5] is a digital proof of authenticity stored on Aura Blockchain. This ensures that each diamond's unique characteristics, provenance, and ownership remain immutable and verifiable.

To transfer an LV Diamonds Certificate [6], the current owner must initiate the process through their My LV Account under the "My Certificates" section. The transfer requires entering the new owner's email and setting a secret keyword. To successfully complete the transfer, the new owner must enter the secret keyword and the diamond's individual number. Once activated, the certificate is assigned to the new owner's account and removed from the previous owner's records.

Pros.

- **Sustainable Blockchain:** Aura utilizes a Proof of Authority (PoA) consensus mechanism, which is far more energy-efficient than Proof of Work (PoW). This makes it a low-energy alternative suitable for large-scale adoption in luxury markets.

Cons.

- **No Physical Link Between Certificate and Diamond:** The certificate itself is secured on the blockchain, but there is no direct physical connection between the certificate and the actual diamond. This means that while a certificate may be genuine, the physical stone could have been replaced with a counterfeit.
- **Private Blockchain – Limited Consumer Transparency:** Since Aura is a closed, permissioned blockchain, consumers cannot independently verify product details on-chain without brand approval. Unlike public blockchains, where anyone can access transaction records, Aura restricts visibility.
- **Restricted Access to Product Certificates:** Customers only see a controlled, brand-approved representation of the blockchain record. This "mirror" approach limits transparency, as buyers cannot access direct on-chain verification to cross-check ownership or transaction history.

7.1.2 Comparison of Ethereum and Polygon in Cost Efficiency and Transaction Speed for Blockchain Operations. In the paper "A Comparative Study of Ethereum and Polygon for Implementing NFT-Based Certification Systems" by M. T. Aung and N. N. M. Thein [7], the authors compare the cost efficiency and transaction speed of Ethereum and Polygon.

Regarding cost efficiency, minting 100 NFTs on Ethereum costs approximately $74.64, whereas the same process on Polygon costs only $0.20. This significant difference is primarily due to Ethereum's higher gas fees, which increase with network congestion.

As for transaction speed, Polygon can process up to 7,000 transactions per second (TPS), while Ethereum is limited to just 22.5 TPS. This makes Polygon a more scalable solution, particularly for applications requiring frequent and high-volume transactions.

Since Polygon outperforms Ethereum in cost efficiency and transaction speed, we refer to the system model from another related paper that uses Polygon to address counterfeiting and tampering in judicial documents as a reference.

7.1.3 Decentralized Model to Protect Digital Evidence via Smart Contracts Using Layer 2 Polygon Blockchain. In this paper, they applied blockchain technology to the judicial system for managing digital evidence. [8] By leveraging blockchain, it reduces the risk of tampering and intentional deletion of digital evidence. We believe that their proposed application architecture aligns well with our research topic. Unlike the current LV Diamonds Certificate solution, which relies on a digital certificate activated via email and a keyword, this approach utilizes the Polygon public blockchain, allowing all customers to directly read and access data on-chain. This enhances transparency, security, and accessibility compared to traditional authentication methods. Moreover, instead of storing legal documents, we can record handbag manufacturing data (e.g., year, model, leather type, collection season) on IPFS, allowing consumers to verify authenticity.

System Model. The Polygon Blockchain (Mumbai Testnet) serves as a testing environment that mirrors the Polygon Mainnet, allowing researchers to conduct experiments using free testnet tokens. Smart contracts manage access permissions for authorized judicial personnel, such as police officers, lawyers, and prosecutors. Instead of storing full documents on the blockchain, only the CID (content identifier) of each judicial record is recorded on IPFS (InterPlanetary File System), a decentralized storage network. This method reduces the data load on the blockchain while ensuring secure and verifiable evidence management.

Pros.

- **Evidence Integrity:** The blockchain's immutability ensures that evidence cannot be tampered with after storage.
- **Decentralized Storage** Reduces the data load on the blockchain by storing only the CID (Content Identifier) on-chain while keeping the actual data in a decentralized off-chain storage system.

Cons.

- **Front-End Exploits:** DApps rely on front-end interfaces, they are vulnerable to traditional web security threats like XSS, CSRF, and phishing attacks.

7.2 Advantages of Our Approach

Our approach integrates Polygon's Layer 2 blockchain and Ed25519 digital signatures to strengthen decentralization, improve the link between NFC tags and products, and reduce the risk of counterfeiting. At the same time, we develop a dApp that allows consumers to independently verify products, ensuring that authentication is no longer solely controlled by brands.

Current solutions, such as the blockchain developed by Aura Consortium, offer authentication techniques but face problems with efficiency, security, and transparency. The biggest drawback is its centralized nature—only approved luxury brands can validate transactions and manage authentication records. Because of this, consumers are less transparent and are forced to rely on brands for validation. By utilizing Polygon's public blockchain, our technology resolves this issue and makes all registered product data publicly verifiable. Consumers can independently check product provenance without relying on brand approval, ensuring trustless authentication.

Another significant problem with Aura's digital certificate architecture, like the LV Diamonds Certificate, is that there is no direct physical-to-digital connection. The certificate is not cryptographically linked to the product, thus counterfeiters can place real certificates on phony goods. Our system solves this by embedding Ed25519-signed NFC tags inside products. Each tag contains a manufacturer-signed unique identifier, making it impossible to clone or replace without detection. The cryptographic signature ensures only genuine products pass verification, creating a tamper-proof authentication mechanism.

We also improve blockchain storage efficiency. Instead of storing full product records on-chain, we store only essential metadata and decentralized storage references. This reduces blockchain storage costs while keeping records immutable and easily accessible. Even as product registrations grow, our system remains scalable and cost-effective.

Finally, user adoption is a key challenge. Many solutions require brands to manage complex blockchain interactions, making implementation difficult for those without technical expertise. We solve this by providing a dedicated platform for brands, allowing them to onboard products without blockchain knowledge. For consumers, our user-friendly dApp enables instant authentication via NFC scanning, keeping verification simple and accessible.

7.2.1 Comparison with Non-Blockchain and NFT-Based Methods. Traditional anti-counterfeiting approaches such as QR codes, barcode stickers, or engraved serial numbers offer a low-cost solution, but they are prone to duplication and tampering. Most of these methods lack cryptographic verification or immutable records, making them easy targets for forgery.

Meanwhile, some blockchain-based authentication systems rely on NFTs to represent product ownership. Although NFTs improve traceability, they do not guarantee the authenticity of the physical product, as there is often no secure link between the digital certificate and the real-world item. Without a trusted tag or hardware-secured identifier, NFT records can be transferred to counterfeit goods.

Our system addresses this gap by using cryptographic signatures (Ed25519) embedded in NFC tags, which are independently verifiable using data from the public blockchain.

Table 4: Comparison of Authentication Methods

Method	Decentralized	Signature	Open Query	Physical Link
QR Code	No	No	Yes	No
Aura Blockchain	No	No	No	No
NFT-based System	Yes	No	Yes	No
Our Approach	Yes	Yes	Yes	Yes

8 PROJECT TIMELINE AND EXPECTED MILESTONES

To ensure timely completion, the project will follow the schedule outlined in Table 5.

Table 5: Project Timeline

Date	Milestone
February 2025	Completion of system architecture and cryptographic selection
March 2025	Development of smart contract prototype and blockchain integration
April 2025	Implementation of NFC-based digital signature verification and consumer authentication system
May 2025	Completion of user-facing application and final security/performance testing

9 CONCLUSION

In this project, we designed and implemented a blockchain-based authentication system that leverages Polygon zkEVM, IPFS, and Ed25519 digital signatures to address the persistent issue of counterfeiting in the luxury goods market. By combining decentralized infrastructure with cryptographic guarantees, our system allows consumers to independently verify the authenticity of products via NFC scanning without relying on brand-controlled servers.

Our initial implementation demonstrates that the proposed approach is technically feasible and practically efficient. Performance evaluation across different data sizes confirms that storage on IPFS is scalable and cost-effective, while smart contract operations remain gas-stable and predictable. Additionally, our signature verification experiment showed a 90% detection rate for tampered tags, indicating strong forgery resistance even in simulated scenarios.

Unlike traditional solutions such as the Aura Blockchain, which depend on centralized control and lack a direct physical-to-digital link, our method cryptographically binds each product to its metadata and identity through digitally signed NFC tags. This ensures not only data immutability but also hardware-level verifiability.

For the next phase, we aim to extend the implementation with actual NFC tag hardware integration, improve the user experience of the consumer dApp, and explore mechanisms for securely managing key rotation and revocation. We also plan to investigate interoperability with other EVM-compatible chains and explore additional use cases, such as art collectibles or digital twins for luxury assets.

Overall, our approach offers a transparent, decentralized, and scalable solution for building consumer trust and combatting counterfeiting in high-value goods supply chains.

REFERENCES

[1] Forbes. (2025). "The Global Counterfeit Market." Available: https://www.forbes.com

[2] V. Clincy and H. Shahriar, "Blockchain Development Platform Comparison," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, Milwaukee, WI, USA, 2019, pp. 922–923. Available: https://doi.org/10.1109/COMPSAC.2019.00142.

[3] Y. Liu, X. Huang, and Y. Yang, "Cryptographic Accelerators for Digital Signature Based on Ed25519," in *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1251–1263, Aug. 2021. Available: https://ieeexplore.ieee.org/document/9437467

[4] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "The Provable Security of Ed25519: Theory and Practice," in *Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 2021, pp. 1085–1102. Available: https://ieeexplore.ieee.org/document/9519456

[5] Louis Vuitton. Aura Consortium Blockchain & The LV Diamond Certificate. Available: https://us.louisvuitton.com/eng-us/faq/services/aura-consortium-blockchain-the-lv-diamond-certificate.

[6] Louis Vuitton. LV Diamonds Certificate Transfer. Available: https://us.louisvuitton.com/eng-us/faq/services/lv-diamonds-certificate-transfer.

[7] M. T. Aung and N. N. M. Thein, "A Comparative Study of Ethereum and Polygon for Implementing NFT-Based Certification Systems," in *2024 5th International Conference on Advanced Information Technologies (ICAIT)*, Yangon, Myanmar, 2024, pp. 1–6. Available: https://doi.org/10.1109/ICAIT65209.2024.10754936.

[8] S. K. Rana *et al.*, "Decentralized Model to Protect Digital Evidence via Smart Contracts Using Layer 2 Polygon Blockchain," in *IEEE Access*, vol. 11, pp. 83289–83300, 2023. Available: https://doi.org/10.1109/ACCESS.2023.3302771.

APPENDIX A: SMART CONTRACT IMPLEMENTATION

The following is an excerpt of the deployed smart contract written in Solidity v0.8.20. It supports brand registration, product registration, deactivation, and data retrieval.

```
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/access/Ownable.sol";

contract ProductRegistry is Ownable {
    struct Product {
        string ipfsCid;
        string publicKey;
        uint256 registrationTime;
        bool isActive;
    }

    mapping(string => Product) public products;
    mapping(string => bool) public brandIds;

    function registerBrand(string memory brandId) public
         onlyOwner {
        require(!brandIds[brandId], "Brand already 
            registered");
        brandIds[brandId] = true;
    }

    function registerProduct(
        string memory productId,
        string memory ipfsCid,
        string memory publicKey
    ) public onlyOwner {
        require(!products[productId].isActive, "Already 
            registered");
        products[productId] = Product(ipfsCid, publicKey,
             block.timestamp, true);
    }

    function deactivateProduct(string memory productId)
         public onlyOwner {
        require(products[productId].isActive, "Not active
            ");
        products[productId].isActive = false;
    }

    function getProduct(string memory productId) public
         view returns (
        string memory, string memory, uint256, bool
    ) {
        Product memory p = products[productId];
        require(p.isActive, "Not found");
        return (p.ipfsCid, p.publicKey, p.
            registrationTime, p.isActive);
    }
}
```

APPENDIX B: BACKEND SERVER ARCHITECTURE AND API IMPLEMENTATION

To enable integration between the frontend dApp, IPFS storage, and zkEVM smart contracts, we developed a backend system using Node.js and Express. The server provides RESTful endpoints to manage product registration, brand onboarding, IPFS integration, and Ed25519 signature generation and verification. This implementation bridges the frontend interface with decentralized infrastructure.

The backend consists of the following modules:

- **IPFS Integration**: Uses the Pinata SDK to pin product metadata (JSON and image files) and retrieve the resulting CID.
- **Smart Contract Interaction**: Uses ethers.js to connect to the Cardona zkEVM testnet, allowing registration and retrieval of on-chain product data.
- **Signature Utilities**: Ed25519 keys and digital signatures are generated and verified using `tweetnacl`, wrapped in reusable helper functions.
- **Validation and Error Handling**: All incoming product data is validated before submission to ensure structural and logical correctness.
- **Logging and Diagnostics**: All API calls are logged with timestamps and diagnostic information to support debugging and transparency.

Brand Registration Endpoint

```
app.post('/api/brands', async (req, res) => {
    const { brandId } = req.body;
    try {
        const tx = await contract.registerBrand(brandId);
        const receipt = await tx.wait();
        res.json({
            success: true,
            transactionHash: tx.hash,
            gasUsed: receipt.gasUsed.toString()
        });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});
```

Listing 1: POST /api/brands - Register a brand

Product Registration Endpoint

Product metadata and associated image are uploaded to IPFS, signed using Ed25519, and the public key is stored on-chain.

```
app.post('/api/products', async (req, res) => {
    const { productId, brandId, serialNumber, ipfsCid } =
        req.body;
    const { publicKey, privateKey } = generateKeyPair();
    const message = JSON.stringify({ productId, brandId,
        serialNumber });
    const signature = await sign(message, privateKey);
    const tx = await contract.registerProduct(productId,
        ipfsCid, publicKey);
    const receipt = await tx.wait();
    res.json({
        success: true,
        transactionHash: tx.hash,
        signature,
        publicKey,
        gasUsed: receipt.gasUsed.toString()
    });
});
```

Listing 2: POST /api/products - Register a new product

Product Verification Endpoint

This endpoint receives tag data and performs a full verification against blockchain and IPFS data.

```
app.post('/api/products/:productId/verify', async (req,
     res) => {
    const { productId } = req.params;
    const { brandId, serialNumber, signature,
        tagPublicKey } = req.body;

    const productInfo = await contract.getProduct(
        productId);
    const productData = await fetchIPFS(productInfo[0]);

    const message = JSON.stringify({ productId, brandId,
        serialNumber });
    const signatureValid = await verifySignature(message,
         signature, tagPublicKey);

    res.json({
        verified: signatureValid,
        ipfsCid: productInfo[0],
        productData
    });
});
```

Listing 3: POST /api/products/:productId/verify - Product verification

Utility: Signature Generation and Validation

The following helper functions are used to produce and verify Ed25519 digital signatures using tweetnacl.

```
const nacl = require('tweetnacl');
const { encode, decode } = require('base64-arraybuffer');

function generateKeyPair() {
    const kp = nacl.sign.keyPair();
    return {
        publicKey: encode(kp.publicKey),
        privateKey: encode(kp.secretKey)
    };
}

function sign(message, privateKey) {
    const msgUint8 = Buffer.from(message, 'utf8');
    const sk = decode(privateKey);
    const sig = nacl.sign.detached(msgUint8, sk);
    return encode(sig);
}

function verifySignature(message, signature, publicKey) {
    const msgUint8 = Buffer.from(message, 'utf8');
    return nacl.sign.detached.verify(
        msgUint8,
        decode(signature),
        decode(publicKey)
    );
}
```

Listing 4: Signature generation and verification using tweetnacl

APPENDIX C: AUTHOR CONTRIBUTIONS

This project was completed collaboratively by both authors. The division of responsibilities is as follows:

- **Hsuan-Chi Chang (50%)**: Led the design of the system architecture, implemented the smart contract and back-end verification API, developed the consumer-facing dApp prototype, and conducted system evaluation experiments (IPFS, zkEVM, accuracy of the signature).
- **Wei-Ju Li (50%)**: Contributed to collected and compared related works, cryptographic design, conducted literature review and competitive analysis, supported UI design and implementation of Ed25519 verification, and led technical writing and editing.

Both authors collaborated on system planning, problem definition, and report presentation.