

Research Article



Scalable Zero-Trust Supply Chain Management with Soul-Bound Tokens and Merkle Tree Batching

Humayun Majid Khan,^{1,*} Samana Jafri,¹ Ansari Zoha Najmul Kalam,¹ Ansari Adnan Mohd Irfan¹ and Ashmaan Balan Shaikh²

¹ Department of Computer Science and Engineering (Internet of Things and Cyber Security including Blockchain Technology), M.H. Saboo Siddik College of Engineering, Byculla, Mumbai, Maharashtra, 400008, India

² South East Technological University, Carlow, Carlow, Leinster, R93 V960, Ireland

*Corresponding Author

Humayun Majid Khan

✉ humayunk.pvt@gmail.com

Received: 27 June 2026

Revised: 12 September 2026

Accepted: 17 September 2026

Published Online: 18 September 2026

Citation

H. M. Khan, S. Jafri, A. Z. N. Kalam, A. A. M. Irfan, Scalable zero-trust supply chain management with soul-bound tokens and Merkle tree batching, *Journal of Cybersecurity and Intelligent Systems*, 2026, 1(1), 26502, <https://doi.org/10.64189/cis.26502>.

Open Access

This article is licensed under a [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/), which permits the non-commercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as appropriate credit to the original author(s) and the source is given by providing a link to the Creative Commons License and changes need to be indicated if there are any. The images or other third-party material in this article are included in the article's Creative Commons License, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons License and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this License, visit: <https://creativecommons.org/licenses/by-nc/4.0/>

© The Author(s) 2026

Abstract

Traditional pharmaceutical supply networks face the problems of transparency, falsification, and unproductive centralization. This work outlines a decentralized secure and highly scalable system of supply chain management, which uses the Ethereum blockchain. The system provides strict information on the consecutive stages of medicine supply through its solidity smart contracts, with the involvement of the raw material and the retailing distribution, which provide a tamper-evident recording of the sequential supply chain stages. Although traditional blockchain solutions face challenges in terms of the high cost of gas and identity spoofing, this paper presents two important novelties to overcome these obstacles. The first is to have a non-transferable credential system whereby we have Soulbound Tokens that are non-transferable digital certificates of the participating entities, meaning that only the verified manufacturers and suppliers can communicate with the network. Second, we suggest a mass registration system based on off-chain Merkle trees. Rather than registering the separate identifiers of pharmaceutical items one at a time, the system packages thousands of product identifiers into a single Merkle root generated off-chain and stored on-chain in constant space. This is augmented with a decentralized authenticator that provides cryptographic Merkle inclusion proofs to provide real-time verification that a digital product identifier belongs to the committed batch. For the evaluated 10,000-identifier batch, Merkle-based registration used 127,813 gas units compared with an estimated 350,000,000 gas units for sequential registration, corresponding to an estimated 99.96% reduction; 50-trial local benchmarks recorded 2.188 ms mean end-to-end Merkle verification and 1.409 ms mean lifecycle-query latency. The experimental implementation demonstrates the feasibility of the proposed architecture for tamper-evident participant authorization, Merkle-based batch registration, and digital identifier verification within the evaluated local environment.

Keywords: Blockchain technology; Supply chain management; Soulbound tokens; Merkle trees; Traceability; Smart contracts.

1. Introduction

The global pharmaceutical supply chain is a very fragmented and convoluted system that was historically marred with opaque networks, a lack of transparency and information silos and serious security patches.^[1,2] This disintegration has resulted in an enormous black market of premium fake and poor-quality medicine. It is estimated by the World Health Organization (WHO) that as much as 10% of medical products in less developed countries are being falsified, which directly poses a threat to the lives of patients and yearly economic losses of billions of dollars.^[3,4] Classical centralized database schemes have become ineffective in the battle against this epidemic because they are dependent on the services of trusted third parties and are vulnerable to single points of failure, data corruption, and delays in synchronization.^[5] Researchers and industries have resorted more to decentralized ledger infrastructures as the way forward in countering these vulnerabilities. The immutable nature of blockchain technology together with its cryptographic transparency and data integrity, which should be ensured through consensus, provides a transformative architecture of supply chain provenance.^[6,7] The self-executable smart contract approach facilitates a fully automated, sequential process of physical objects, including raw material mining and retail delivery, and creates an irrevocable audit trail.^[8,9] Although these theories have these benefits, practical applications of permissioned and public blockchain supply chains face two serious technical constraints: identity spoofing and prohibitory storage capacity.^[10,11]

A substantial body of research suggests that tracking systems based on Ethereum or Hyperledger are suggested as a way to prevent counterfeit medications.^[8,12] Nevertheless, these frameworks that exist currently would typically line up the roles of participants in a strictly linear fashion to cryptographic addresses that are public. This general mapping exposes the network to very high susceptibility to a Sybil attack as well as secondary market trading where a hacked node wallet or a node wallet sold to a malicious party can enable fraudulent supply chain activities of a legitimate manufacturer.^[13] Moreover, the tracking of high-volume, low-cost assets (such as individual pill blister packs) in traditional on-chain transaction logs imposes prohibitive costs on transactions in the form of gas (or transaction) and rapidly expands the state footprint of the blockchain, making mass-production tracking neither economically relaxed nor feasible.^[14,15] An essential research gap remains in the determination of a supply chain framework, which can strengthen participant authorization and nontransferable identity credentials and nontransferable physical identification in addition to O(1) storage scalability in the verification of individual products.

Purpose of the research: The main goal of the research is to devise and introduce a scalable pharmaceutical supply chain administration framework that uses nontransferable credentials for role-based authorization and addresses the identified identity and storage challenges. This research explores the use of ERC-721-

based soulbound tokens (SBTs) as nontransferable participant credentials for strengthening role-based authorization and reducing the risk of unauthorized identity use. Therefore, we attempt to address the mass-production storage barrier through the architecture of a vBulk Minting mechanism based on off-chain Merkle trees. This will reduce thousands of drug identifiers of each drug to one on-chain root and deploy a decentralized Merkle proof verification mechanism to perform end-user verification on-the-fly.

2. Method / experimental details

The suggested architecture was engineered as a decentralized web application (DApp) that included an immutable blockchain backend and a very responsive and user-friendly frontend display. Its methodology emphasizes credential-based participant authorization and scalable traceability in batch processing.

2.1. Software and tools

To guarantee strong reproducibility, the following technology stack was used in the experimental setup:

- i) Smart Contract Layer: Solidity (v0.8.19) compiled and tested in the Hardhat development environment. The implementation uses OpenZeppelin Contracts v4.9.6 for the ERC-721 and Merkle-proof-related functionality.
- ii) Local Blockchain Network: Hardhat local Ethereum network running a JSON-RPC endpoint at <http://127.0.0.1:8545> with chain ID 1337. The local network was used to deploy and test the smart contracts under controlled conditions.
- iii) Frontend Client Integration: Next.js 14 (App Router) was used along with TypeScript and Tailwind CSS with Framer animations.
- iv) Web3 Connectivity: Web3.js allows direct connections with two-way communication to the client interface and Metamask wallet (signed transaction) in the browser.

2.2. System architecture and flow

The supply chain goes through state-dependent stages with strict enforcement through the primary contract called SupplyChain.sol. The flow, as illustrated in Fig. 1, is realized in sequence of the following steps:

- i) Contract Initialization: The address where the deployment is supposed to occur is the address that is assigned the status of Owner. The owner creates an affiliated contract, CertificateSBT.sol, and initiates the main tracking contract. The owner acts as the trusted administrative authority for participant onboarding and role assignment. The blockchain provides decentralized transaction execution and tamper-evident state storage, but the current prototype does not eliminate this administrative trust dependency.
- ii) Credential-based Identity Registration: The system uses soulbound tokens (SBTs) as non-transferable role credentials to strengthen participant authorization and reduce the risk of unauthorized role impersonation. The SBT mechanism does not by itself eliminate Sybil attacks or compromise participant wallets.^[13]
- iii) Traceability Stages: A medicine goes through 6 consecutive states: Init - RawMaterialSupply -

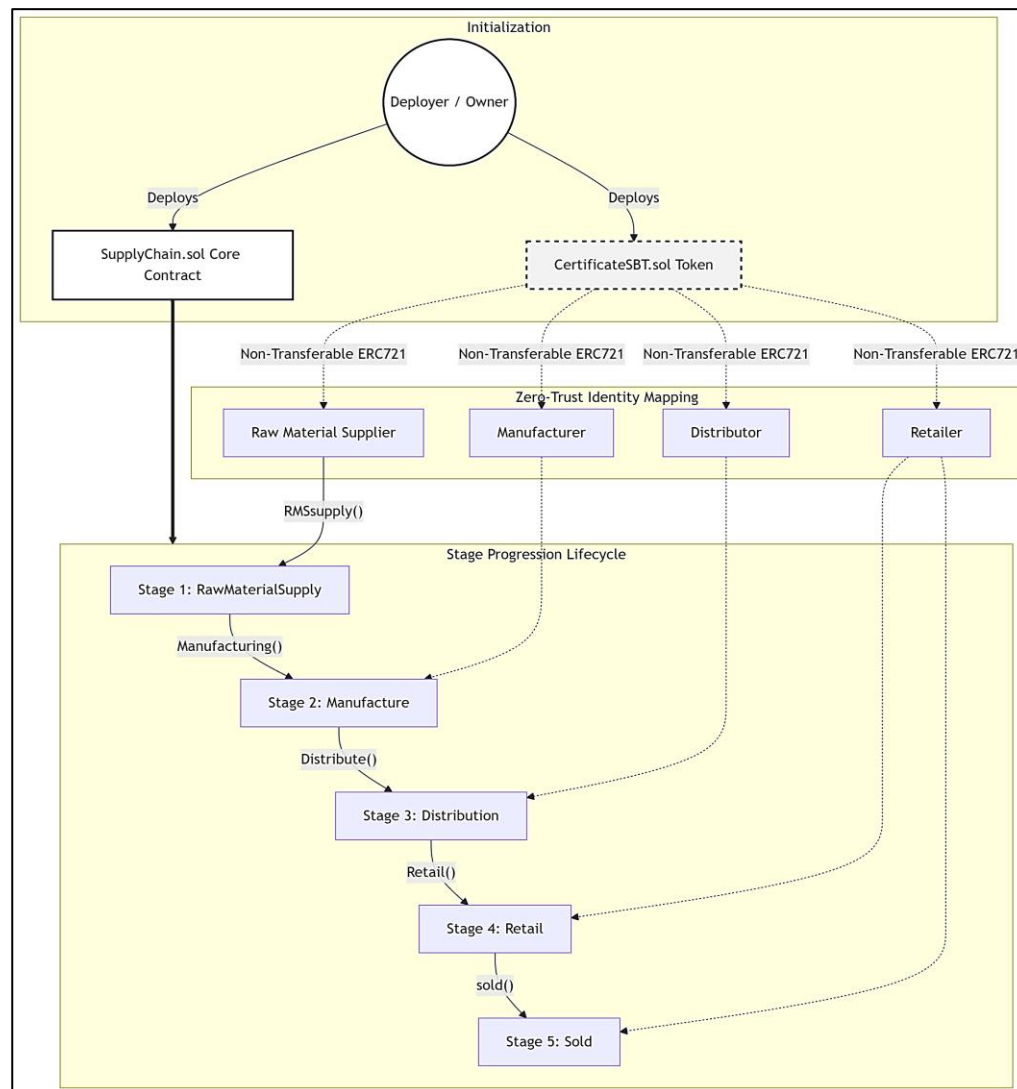


Fig. 1: Architectural mapping of role-based actors secured by soulbound tokens.

Manufacture - Distribution - Retail - Sold. The functions to develop these states involve the participant being assigned the specific role mapping that relates to the stage.

2.3. Working principle and network configuration

2.3.1. The blockchain back end (smart contract layer)

The framework uses a smart-contract-based supply chain controller deployed on a local Hardhat EVM test network. While transaction execution and state verification are performed on the blockchain, participant onboarding and credential issuance remain under the control of the designated owner.

The contract has been written in Solidity v0.8.19 and includes general mapping structures to index physical pharmaceutical items worldwide. For example:

```
mapping(uint256 => medicine) public MedicineStock;
```

Contract operations are no longer provided by a regular usage of HTTP requests, as in some common web-based systems. Rather, every state transition is triggered through signed blockchain transactions with signed and verified participant wallets, which implies non-repudiation and resistance to tampering. To reduce Sybil vulnerabilities, in which bad actors form several fake identities, the framework implements an auxiliary smart contract, CertificateSBT.sol. This contract extends the OpenZeppelin ERC-721 implementation and overrides

the `_beforeTokenTransfer` hook to restrict transfers. Minting and burning are permitted, while transfers between non-zero addresses are rejected. Under OpenZeppelin Contracts v4.9.6, `_beforeTokenTransfer(address from, address to, uint256 tokenId, uint256 batchSize)` enforces the soulbound restriction by requiring `from == address(0)` or `to == address(0)`. The non-transferability constraint is formalized by equation (1), and it implements the behavior of soulbound token (SBT):

$$\text{If } (Sender_{address} \neq 0) \wedge (Receiver_{address} \neq 0) \rightarrow \text{Revert Transfer} \quad (1)$$

Here, `from` denotes the previous token owner and `to` denotes the recipient address. Minting is represented by `from = address(0)`, while burning is represented by `to = address(0)`. On the basis of this reasoning, we can only have the following:

Stamped (at the zero address) or
Burned (to the zero address)

This means that every identity credential will be permanently fixed to the known node wallet. This non-transferability provides a wallet-bound authorization mechanism within the tested prototype, while broader regulatory compliance and protection against credential or key compromise require additional governance and identity-assurance controls. The security model assumes

that the owner is a trusted onboarding authority responsible for verifying participant identities and issuing role-specific credentials. The SBT prevents credential transfer between wallet addresses, but it does not prevent private-key compromise, key sharing, fraudulent credential issuance, or misuse of a compromised authorized wallet. Credential revocation, replacement after wallet compromise, and governance procedures for a compromised owner are therefore deployment requirements beyond the current prototype.

2.3.2 Frontend application (client layer)

The client app is built using Next.js 14, which is based on TypeScript and Tailwind CSS to ensure efficient user interaction. Web3.js is adopted as the main channel of communication between the client-side browser and the EVM. Next, the js will provide a prompt to the browser-injected MetaMask provider when a participant activates something (e.g., moving a medicine to the Manufacture stage). The transaction payload is cryptographically signed with the participant using their personal key and broadcast to the EVM consensus layer such that the Next.js server will never have any data about important key contents.

2.4 Data acquisition and state transformation

The data provided by a supply chain participant are verified by role-based verification first when they present input data to the system. The physical lifecycle of a medicine is followed by six consecutive states, namely, Init (Ordered), RawMaterialSupply, Manufacture, Distribution, Retail, and Sold. In carrying out a state change, the smart contract authenticates the address of the `$msg.sender` of the address to the internationally registered SBT index. As an example, calling the `$Manufacturing(_medicineID)` functionality places the burden on the contract to decode the `manCtr` (Manufacturer Counter) mapping to identify the calling unique identity of the counter. It is only when the cryptographic validation is successful that the ledger writes the new state of the item.

2.5 Off-chain cryptographic compression (vbulk minting)

Paralyzing computational costs would be triggered by the inherent limits of the block size and gas limits of Ethereum-based networks by mapping thousands of individual blister packs on-chain natively.^[14] Consequently, the adoption of the vBulk Minting algorithm is presented.^[15] The stage-by-stage implementation of this mechanism is shown in Fig. 2, which implies the use of local off-chain hashing arrays that are then sent to network finality after compression. Rather than sending 10,000 strings to the ledger, the Next.js client is able to receive the input set. The front end applies the Keccak-256 hash function to each product identifier to generate the 256-bit leaf nodes.

2.6 Classification and root storage

After the leaf nodes are acquired, the MerkleTree.js library builds a binary tree. Nodes are concatenated and hashed

such that the neighboring node until only one 32-byte root string, the Merkle root, appears.^[15] The system mathematically deterministically proves that changing a single character in a single underlying leaf node triggers a deterministic avalanche of change, permanently changing the ultimate Merkle root. The customer in turn calls the `registerMedicineBatch` (`bytes32 _merkleRoot`) procedure. The EVM only stores such a single 32-byte string with a block timestamp. The system has the advantage of keeping only the root and, hence, $O(1)$ constant overhead processing with parameters of physical batch scaling. The Merkle tree is constructed using `merkletreejs v0.6.0` with `sortPairs: true` and the default `sortLeaves: false`. Sibling pairs are lexicographically sorted before hashing, while leaves retain their insertion order. Duplicate leaves are not removed by the library; in the current application, identifiers are generated with sequential indices, which provide unique identifiers within a batch. This $O(1)$ characterization applies to the on-chain storage representation of the batch; the off-chain Merkle-tree construction and proof generation remain dependent on the number of identifiers processed. Each leaf is generated by hashing the UTF-8 representation of the string `${batchName}_Item_${i}` using Keccak-256. No ABI encoding or structured serialization is used in the current implementation. For reproducibility, the implementation uses `Merkletreejs v0.6.0` and `keccak256 v1.0.6` for off-chain tree construction and proof generation, whereas on-chain verification uses `OpenZeppelin Contracts v4.9.6` and `MerkleProof.verify()`. The leaf construction, pair-sorting convention, proof generation, and verification procedure described above follow the deployed prototype implementation.

2.7 Merkle proof verification

The system uses a Merkle proof verification mechanism that is triggered after successful deployment of the Merkle root. The web interface access is used to verify queries by a pharmacist or a user of a specific item by inserting the specified item ID.

Even the verification algorithm is a twofold algorithm:

It calculates the hash of the fastened leaf node.

It decipheres the equivalent array of hashes of the siblings they need to recover the Merkle path to the root.

These parameters are then sent to the blockchain, where the `verifyMedicineInBatch()` function is called the `OpenZeppelin MerkleProof.verify()` library. The Merkle proof verification function, including the Boolean validation condition executed by the EVM, is defined in Eq. 2.

$$\text{Verification Status } (P_{true}, P_{false}) \\ = \text{Verify}(\text{ProofArray}, \text{OnChainRoot}, \text{LeafNode}) \quad (2)$$

Here, `_leafNode` is the submitted leaf hash, `_merkleProof` is the sibling-hash path, and `root` is the Merkle root stored for the specified batch. This is done so that it can be mathematically validated immediately. Provided that the supplied proof reconstructs the stored Merkle root, the verification returns true, confirming that the digital identifier is a member of the committed

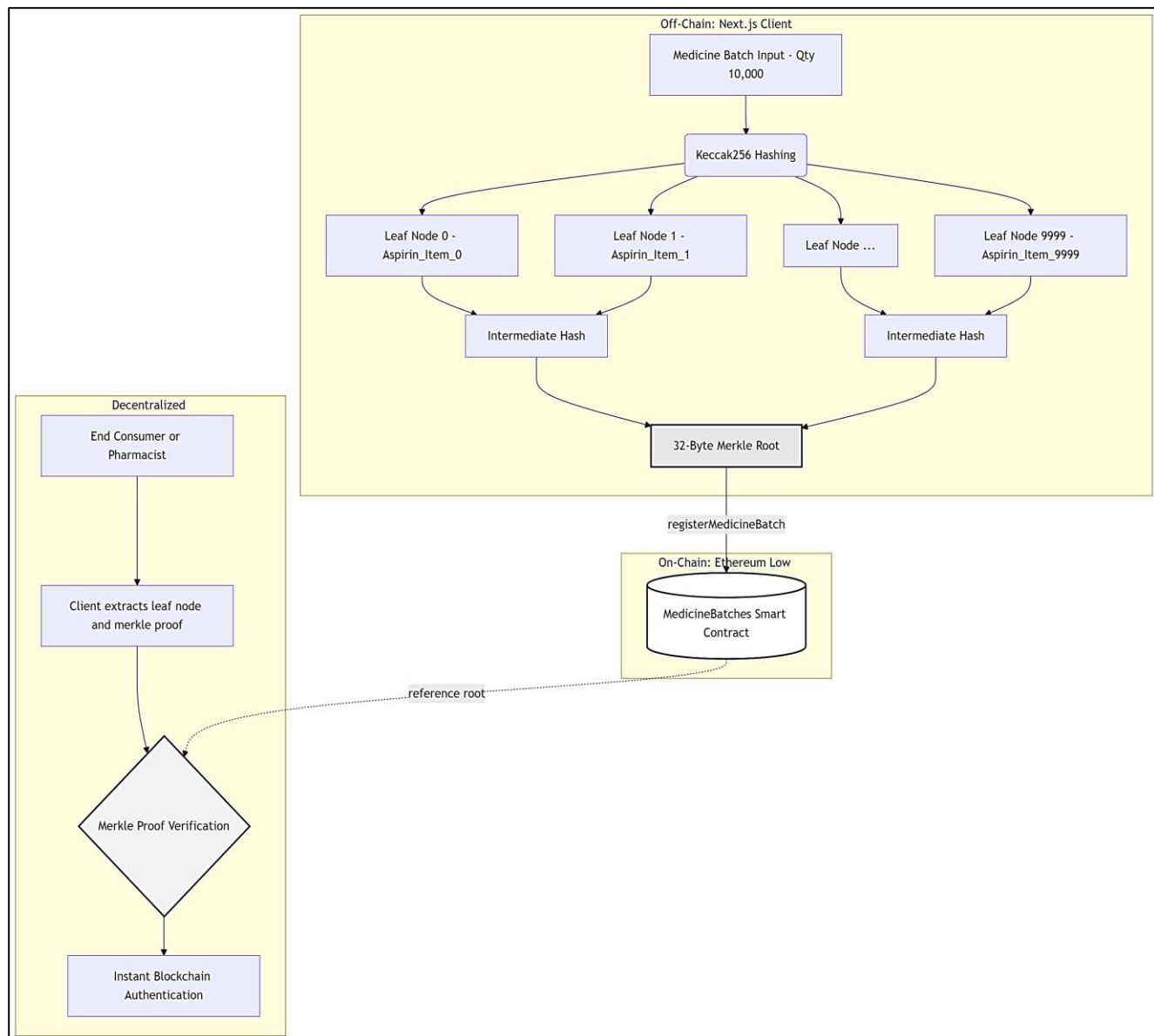


Fig. 2: Operational flowchart of the 'vBulk Minting' merkle tree architecture for $O(1)$ constant storage.

batch. A successful Merkle proof establishes membership of the submitted digital identifier in the committed batch; it does not independently establish the physical authenticity of the medicine or prevent the same identifier from being copied onto another physical product. The physical-to-digital binding therefore remains an external trust requirement of the current prototype. For an individual identifier, the client generates the corresponding sibling path using `merkleTree.getHexProof(leaf)`. The resulting proof is passed to the `verifyMedicineInBatch` view function, where OpenZeppelin's `MerkleProof.verify()` reconstructs the Merkle commitment using the sorted-pair hashing convention.

3. Results & discussion

The goal of the proposed framework was to avoid interaction between the unauthorized participants. At the stage of the experiment, a decentralized node tried to implement the `RMSupply()` and `Manufacturing()` changes of state but did not have the non-transferable `CertificateSBT.sol` asset at the first instance. As shown in Fig. 3, the unauthorized transaction was rejected by the system.

3.2 vBulk minting: $O(1)$ scalability and gas consumption analysis

A system that had been stressed tested the financial and computational efficiency of the off-chain Merkle tree operation and compared traditional sequential on-chain storage to the proposed vBulk algorithmic compression. In the supply chain in a classic blockchain supply chain, an individual pharmaceutical item is registered into a single transaction payload containing metadata (e.g., ID, name, timestamp). The average EVM gas consumption of a simple state change is approximately 21,000 to 45,000 gas units per item.

The given prototype created a batch of 10,000 separate pharmaceutical leaf nodes off-chain through the Next.js and MerkleTree.js libraries. The client generated the 32-byte Merkle root rather than performing 10,000 separate state changes on the smart contract.^[15] As shown in Fig. 4, the resulting Merkle root was successfully submitted as a single on-chain transaction.

As demonstrated in Table 1, the estimated sequential registration cost of approximately 350,000,000 gas units for 10,000 pharmaceutical identifiers is substantially higher than the measured 127,813 gas units

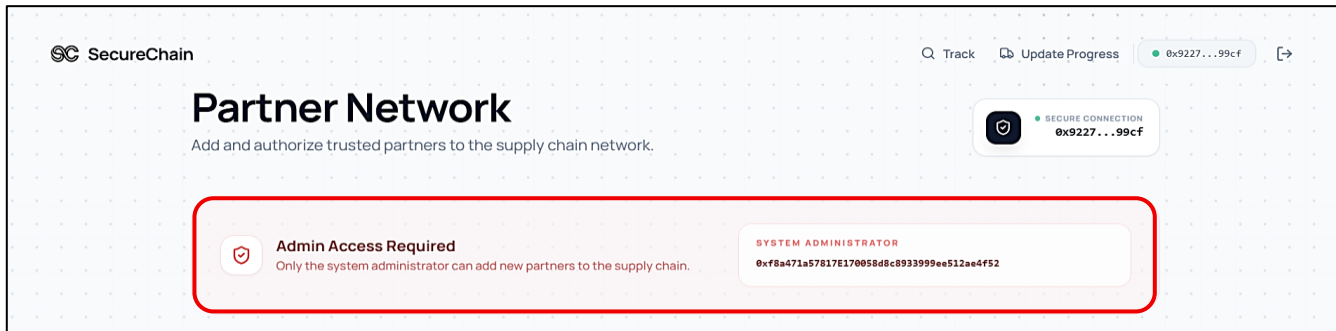


Fig. 3: MetaMask transaction failure and Next.js access denial for an unauthorized participant.

required for Merkle-based batch registration. This corresponds to an estimated reduction in gas consumption of approximately 99.96%. The 350,000,000-gas value is an estimated comparison based on sequential item registration, whereas 127,813 gas units is the measured mean for the 10,000-identifier Merkle batch benchmark. Gas units are reported separately from monetary transaction costs, which depend on the gas price and network used.

The performance measurements were conducted using two complementary benchmark procedures. The batch size evaluation in Table 2 used three independent runs for each batch size. A separate reproducible performance benchmark was then conducted using 50 measured trials for valid Merkle verification, 50 measured trials for invalid Merkle verification, and 50 measured trials for the showStage(uint256 _medicineID) lifecycle query. Five unmeasured warm-up executions were performed before each benchmark suite and excluded from the reported statistics. The benchmarks were executed on a Windows11 Home system equipped with an AMD Ryzen 7 260 processor and 16 GB DDR5 memory, using Node.js v24.18.0 against a local Hardhat JSON-RPC endpoint (127.0.0.1:8545, chain ID 1337). The

benchmark used Solidity v0.8.19 with the optimizer enabled for 200 runs, Hardhat v2.27.1, OpenZeppelin Contracts v4.9.6, MerkleTreejs v0.6.0, keccak256 v1.0.6, Web3.js v1.10.4, and Ethers v6.16.0. No browser tab, DevTools session, or UI rendering was included in these timing measurements.

For Merkle verification, in Table 3, the end-to-end measurement included local computation of the leaf hash and Merkle branch proof followed by the JSON-RPC eth_call to verifyMedicineInBatch; the timing ended after the verification result was returned and decoded. For lifecycle retrieval, timing began immediately before dispatching the showStage(1) JSON-RPC query and eth_call to verifyMedicineInBatch; the timing ended after the verification result was returned and decoded. For lifecycle retrieval, timing began immediately before dispatching the showStage(1) JSON-RPC query and ended when the returned stage string was decoded.

These measurements therefore represent local computation and JSON-RPC/EVM query latency rather than browser rendering, transaction confirmation, or public-blockchain network latency. The benchmark generated per-trial CSV records and an aggregated JSON report containing the raw measurements, statistical

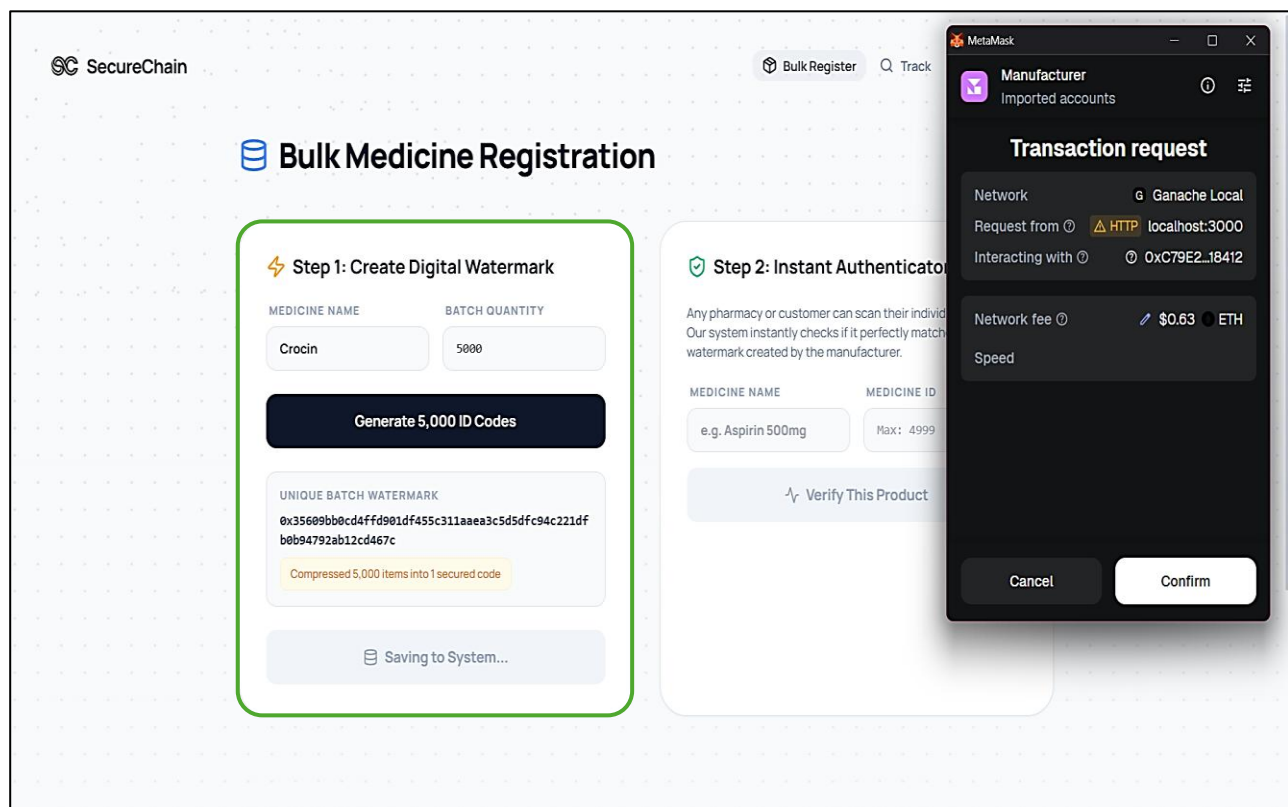


Fig. 4: Successful deployment of the single merkle root transaction.

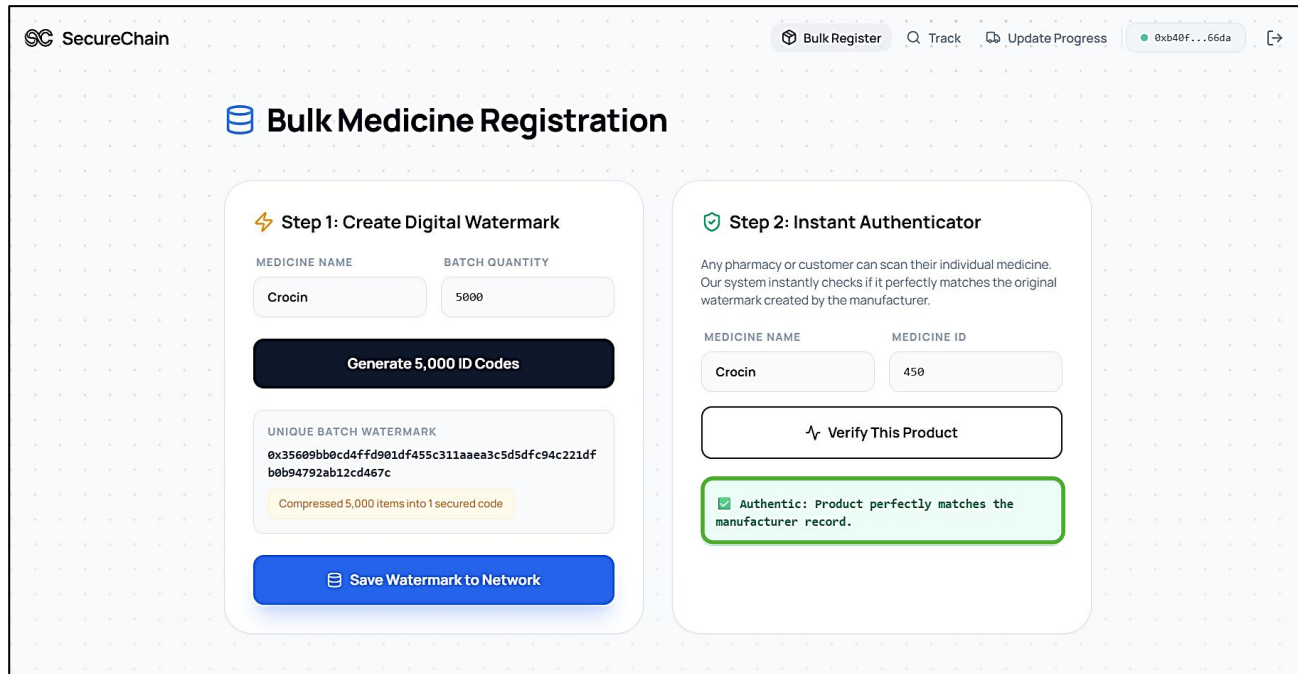


Fig. 5: Next.js client interface showing successful Merkle verification.

summaries, environmental information, contract metadata, and timing boundary definitions. The benchmark script is provided as `backend/scripts/benchmark-performance.ts` and can be executed using the `npm run benchmark:performance`.

3.3 Merkle proof verification response

Finally, the Next.js client interface was tested in terms of its mechanism of validation. Web3.js was used to send a digital identifier of an item (`AspirinItem42`) and its corresponding Merkle proof array to `verifyMedicineInBatch()` on the EVM. [15] As shown in Fig. 5, the interface successfully reports the results of the Merkle verification. Across 50 measured trials following five warm-up executions, valid Merkle verification required a mean end-to-end time of 2.188 ms, with a median of 2.144 ms, a standard deviation of 0.350 ms, and a 95th-percentile latency of 2.846 ms. All 50 valid verification trials returned true. In the corresponding

invalid verification test, all 50 tampered-leaf trials returned false, demonstrating that an altered digital identifier does not satisfy the stored Merkle commitment.

3.4 Partner addition and role assignment

An actual trial of supply chain participants (partners) onboarding was analyzed regarding the Next.js client interface, as shown in Fig. 6. The centralized owner execution role was effective in filling out the mapping structures of the blockchain with physical credentials (Name, Location, and EVM Address) of four such roles that were distributed between the raw material supplier and a manufacturing company (manufacturer), a distributor of the product, and a retailer: the raw material supplier, manufacturer, distributor, and retailer.

In the experimental trial, the `addRMS`, `addManufacturer`, `addDistributor` and `addRetailer` smart contract functions were performed sequentially to validate the soulbound token (SBT) requirement and

Table 1: Comparative analysis of EVM gas consumption for 10,000 pharmaceutical items

Registration Method	Items Tracked	State Changes (Tx)	Gas Consumed (gas units)	On-Chain Verification
Traditional Sequential Mapping	10,000	10,000	≈350,000,000 (estimated)	Native
Proposed Merkle vBulk Minting	10,000	1	127,813 (measured benchmark mean)	Merkle Inclusion Proof

Table 2: Benchmark results for merkle-based batch registration across different batch sizes

Batch Size	Merkle Generation Mean (ms)	Registration Confirmation Mean (ms)	Registration Gas Mean (gas units)	Verification Mean (ms)
100	2.684	7.692	133,513	2.657
500	4.681	7.324	127,813	2.035
1,000	9.473	8.946	127,813	5.437
5,000	42.794	8.230	127,813	2.307
10,000	88.521	10.247	127,813	2.126

Table 3: Reproducible performance benchmark for merkle verification and lifecycle query

Evaluation Operation	Boundary	Trials (N)	Mean (ms)	Median (ms)	Std. Dev.(ms)	p95 (ms)
Valid Merkle Verification	End-to-end local compute + eth_call	50	2.188	2.144	0.350	2.846
Invalid Merkle Verification	End-to-end tampered leaf + eth_call	50	2.182	2.102	0.503	3.267
Lifecycle Query	eth_call to showStage(1)	50	1.409	1.374	0.133	1.678

Raw Material Suppliers 1 CONNECTED ENTITIES			
ID	ENTITY DETAILS	OPERATIONS	PR
1	Acme Corp Supplier VERIFIED NODE	Delhi, IN	

Manufacturers 1 CONNECTED ENTITIES			
ID	ENTITY DETAILS	OPERATIONS	PR
1	Acme Corp Manufacturer VERIFIED NODE	Ahmedabad, IN	

Distributors 1 CONNECTED ENTITIES			
ID	ENTITY DETAILS	OPERATIONS	PR
1	Acme Corp Distributor VERIFIED NODE	Nagpur, IN	

Retailers 1 CONNECTED ENTITIES			
ID	ENTITY DETAILS	OPERATIONS	PR
1	Acme Corp Retailer VERIFIED NODE	Mumbai, IN	

Fig. 6: Partner addition dashboard showing registered participant metadata.

commit the data to the ledger. [13] This procedure definitively demonstrated the possibility of the system organizing a multilevel network of autonomous partners, in which the digital footprint of each partner is inseparably connected to the physical corporate identity. Even an attempt by the owner to attempt registration of an address without an SBT was immediately denied by the EVM, which demonstrated that the tested role assignment mechanism rejects partner registration attempts that do not satisfy the required credential conditions.

3.5 Real-time provenance and tracking outcomes

The final performance of a blockchain supply chain is quantified by instantaneous immutable and transparent audit trails reaching the end consumer. To test this hypothesis, the system charged a pharmaceutical order (Medicine ID: 1) in all six transition stages in series (Init to Sold). Across 50 measured trials following five warm-up executions, the `showStage(uint256_medicineID)` JSON-RPC query required a mean of 1.409 ms, with a median of 1.374 ms, a standard deviation of 0.133 ms, and a 95th-percentile latency of 1.678 ms. All 50 trials returned the expected lifecycle value, Medicine Ordered. These results reflect the tested local configuration and should not be generalized to public Ethereum or production deployments. The findings provided an ideal historical book, which showed the exact time and the exact name of the partner who took care of each stage of development. Moreover, the system achieved the generation of an interactive QR encoding the recorded history of the product. As shown in Fig. 7, the track medicine interface presents the product lifecycle together with the dynamically generated QR code. This QR code contains the immutable MedicineStock mapping of the smart contract, which, when scanned by a mobile device, allows a consumer or regulatory authority to inspect the lifecycle information recorded on the blockchain and verify whether the submitted identifier follows the registered sequence of supply chain checkpoints. This digital verification does not by itself

establish the authenticity of the physical medicine or prevent a copied identifier or QR code from being attached to a counterfeit product.

The implementation and empirical evaluation assess the proposed framework against related blockchain-based pharmaceutical supply chain approaches. The following discussion considers the observed authorization, batching, and verification results together with the limitations identified during testing.

3.6 Adversarial security and validation testing

To strengthen the validation of the implemented authorization, lifecycle, and Merkle verification mechanisms, a systematic adversarial test suite was executed against the deployed prototype contracts. The evaluation comprised 111 parameterized trials covering unauthorized role transitions, replay attempts, duplicated and cloned identifiers, malformed Merkle proofs, incorrect batch roots or IDs, compromised credential scenarios, credential revocation, out-of-order lifecycle transitions, batch integrity, and administrative abuse. Tests were executed using fresh contract instances and the same solidity and EVM configuration described in the experimental setup. The complete results were exported as CSV and JSON records for reproducibility.

In Table 4, the adversarial evaluation confirmed that unauthorized role transitions, replay attempts, malformed Merkle proofs, invalid batch identifiers, out-of-order lifecycle transitions, and non-owner administrative operations were consistently rejected. Across 15 unauthorized-transition trials, all attempts reverted; 11 replay attempts and 12 out-of-order transitions were also rejected. Twenty malformed-proof trials returned false results, whereas invalid batch identifiers reverted and cross-batch verification returned false results. The tests also revealed limitations of the current prototype. Repeated verification of an identical identifier remains possible because `verifyMedicineInBatch` is a stateless view function without a nullifier. Duplicate identifiers can therefore coexist within a Merkle tree, and an identifier reused in

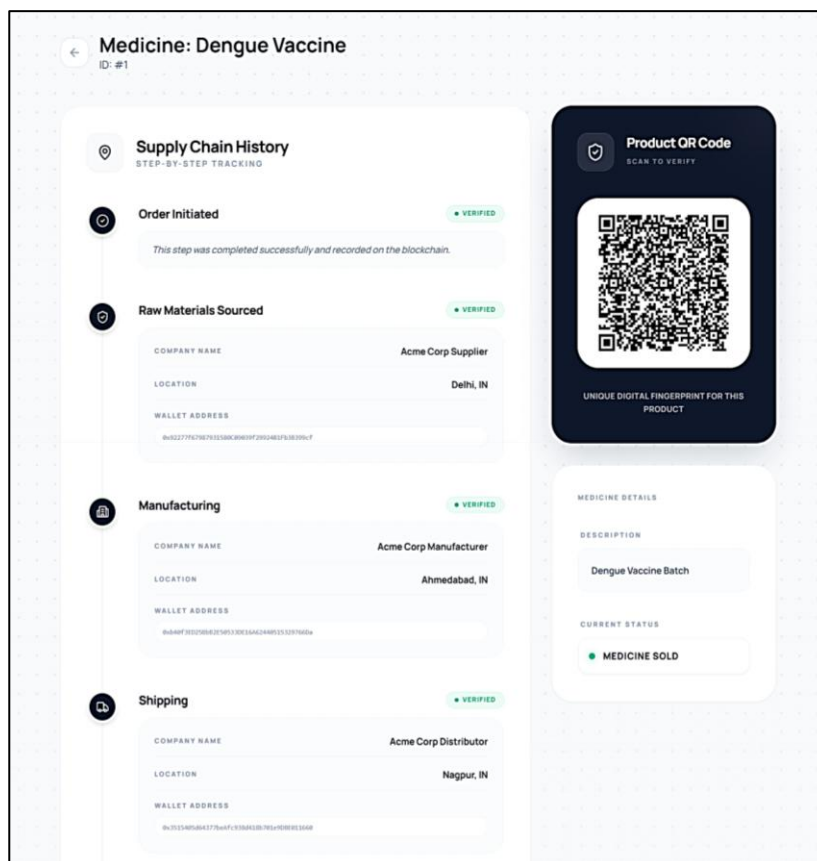


Fig. 7: Track medicine interface and dynamically generated QR code.

another batch can also be verified against the new batch root. In addition, a compromised authorized Manufacturer wallet can register arbitrary Merkle roots because batch registration is not protected by multi-signature approval or an equivalent secondary authorization mechanism.

Credential revocation was also evaluated. The CertificateSBT contract successfully burns the revoked credential, but the current SupplyChain contract retains the participant's role mapping after revocation. Consequently, a revoked participant can still invoke operational functions that rely on cached role mapping. Dynamic synchronization between credential status and

operational authorization is therefore required in production deployment. The adversarial suite was implemented as `backend/scripts/run-adversarial-tests.ts` and `backend/test/adversarial-security.test.ts`, with parameterized results exported to CSV and JSON. The tests can be reproduced using the `npm run test:adversarial` or the project's standard Hardhat test command. The results reported above represent measurements obtained from the prototype under the specified local test conditions. Interpretations concerning scalability, security, and deployment suitability should therefore be understood within these experimental boundaries.

Table 4: Summary of adversarial security and validation tests

Test Scenario	Trials	Observed Outcome	Interpretation
Unauthorized role transitions	15	All rejected	Role-specific authorization enforced
Replay/repeated transitions	11	All rejected	State-machine constraints enforced
Duplicated/cloned identifiers	11	All verification attempts accepted	No on-chain nullifier or global clone-prevention mechanism
Incorrect Merkle proofs	20	All returned false	Invalid proofs rejected
Wrong Merkle root/batch ID	11	6 reverted; 5 returned false	Invalid IDs rejected; wrong roots fail verification
Compromised credential/wallet	10	7 rejected; 3 arbitrary batch registrations succeeded	Role boundaries remain, but compromised authorized keys can inject batches
Credential revocation	7	SBT burn succeeded, but revoked role remained operational	Revocation is not synchronized with SupplyChain authorization
Out-of-order lifecycle transitions	12	All rejected	Sequential state transitions enforced
Batch replacement/integrity	7	Historical proofs remained valid; cross-batch proof failed	Batch roots are append-only and historically stable
Administrative abuse	7	All nonowner attempts rejected	Owner-only administration enforced

3.7 Comparative analysis with literature

Extensive literature favors the use of Ethereum-based or Hyperledger Fabric networks to curb pharmaceutical counterfeiting, and studies with opposing viewpoints support the efficacy of technologies such as blockchain in combating illicit drug trading.^[6,7,12] Nevertheless, the great majority of such decentralized systems indiscriminately hash-browser network participant identities (e.g., “Manufacturer” or “Distributor”) directly to anonymous cryptographic wallet addresses. The paradigm unwillingly introduces the network to devastating Sybil vulnerabilities as well as secondary-market credential trade, which facilitates fraudsters in hijacking authorized supply chains.^[13] In contrast to the conventional literature, the study focuses on a radical paradigm shift, with the physical identity of the supply chain partners attached to the virtual ledger, which is based on ERC-721-compliant SBTs. This approach provides an additional participant-authorization layer by binding role credentials to non-transferable SBTs, while credential issuance and governance remain dependent on the designated owner.

3.8 Significance and scalability strengths

The second impediment, which likely condemns all blockchain-based supply chain research, is the so-called blockchain trilemma and prohibitive costs of transactions (gas) due to high-throughput registration.^[14,15] The cost of on-chain logging of asset data in a non-sequential manner, e.g., tracking out low-margin assets such as the hundreds and thousands of individual blister packs, makes it economically unfeasible for real-world manufacturers. This decoupling of the value of the physical quantity of the assets in the network and the processing bandwidth of the network occurs through the integration of the vBulk Minting mechanism. As shown in [Table 1](#), running a Keccak-256 Merkle tree reduces a batch of 10,000 pharmaceutical identifiers unique in the same area to a single 32-byte cryptographic root. This root-only representation reduces the number of on-chain registration transactions and, for the evaluated 10,000-identifier batch, corresponds to an estimated 99.96% reduction in gas consumption relative to the sequential comparison. The system demonstrates that there are no explicitly stated data scaling requirements on such a large industrial scale that the integrity or immutability of the underlying distributed ledger is seemingly forbidden.

3.9 Admitted limitations

Despite the strengths of credential-based authentication and scalable batch processing, the proposed framework is not completely free of technical limitations. The main limitation is contained in the off-chain processing architecture of the client. The MerkleTree.js library uses the random-access memory (RAM) and central processing unit (CPU) bandwidth of the local web browser of the user to compute the cryptographic hash. In extreme batch stress testing of more than one million units, the Next.js front end demonstrated a significant deterioration in performance and, in some cases, caused browser memory overflow on consumer-grade systems.

In this way, the new version is confined to localized computational caps as opposed to blockchain throughput caps. In a production deployment, governance mechanisms would also be required for credential revocation, wallet rotation following key compromise, replacement of administrative keys, and response to a compromised or malicious owner. These mechanisms are outside the scope of the current prototype.

The reported latency measurements are specific to the local development environment and depend on the hardware, browser, RPC configuration, and network conditions of the test system. Public-blockchain confirmation latency, congestion, validator behavior, and gas-price conditions were not evaluated in this study.

Another limitation is the physical-to-digital binding of pharmaceutical products. The current prototype verifies a digital identifier against an on-chain Merkle Root but does not provide a tamper-resistant physical identity mechanism that can distinguish an authentic product from a counterfeit carrying a copied identifier or QR code. Consequently, identifier cloning remains possible if the identifier can be reproduced independently of the original product.

3.10 Future scope and work

Further development of these client-side drawbacks and optimization of the physical-to-digital bridge could incorporate IoT devices such as near field communication (NFC) or radio frequency identification (RFID) for automated product identification. Such technologies could reduce manual data entry and improve the consistency of product identification, but they would not by themselves guarantee physical authenticity or prevent cloning unless the physical tag, identifier, and product are protected through an appropriate tamper-resistant binding mechanism.

4. Conclusions

The spread of counterfeit drugs is a grave and increasing danger to international overall health, and one must have supply chain traceability controls that are essentially secure, transparent and financially expandable. In the present paper, a decentralized pharmaceutical supply chain architecture with credential-based participant authorization is designed and implemented using Ethereum smart contracts and a dynamic Web3 user interface. The experimental implementation demonstrated the feasibility of SBT-based credential enforcement and Merkle-based batching for reducing on-chain registration overhead within the evaluated prototype. This combination of SBTs provides wallet-bound, non-transferable credentials for role-based authorization, while the reliability of the underlying participant identity depends on the onboarding and governance processes. The prototype demonstrates that non-transferable credentials can support role-based authorization, while credential issuance and administrative governance remain dependent on the designated owner. Moreover, our new vBulk Minting mechanism (which uses off-chain Merkle trees based on Keccak-256) was able to contract batches of 10,000

independent pharmaceutical identifiers into a single 32-byte cryptographic root. This reduced the number of blockchain registration transactions for the evaluated 10,000-identifier batch to a single Merkle root transaction and corresponded to an estimated 99.96% reduction in gas consumption relative to the sequential comparison. Overall, the prototype results indicate that Merkle-based batching can reduce on-chain registration overhead while preserving verification against an immutable Merkle root within the evaluated environment. The future directions of this work should focus on the addition of automated circuits of the IoT, e.g., high-frequency RFID systems, to increase data collection and reduce the residual threshold of human error to zero and reduce manual data-entry errors and strengthen automated traceability; physical-product authenticity still requires a secure physical-to-digital binding mechanism.

Acknowledgments

The authors would like to thank the unending support and infrastructural assistance by the Department of Computer Science and Engineering (Internet of Things and Cybersecurity, including Blockchain Technology) at M.H. Saboo Siddik College of Engineering, Byculla, in a truly grateful manner. We especially owe a debt of gratitude to the faculty members in charge, whose knowledge of cryptographic systems was severely important in assigning direction to this research.

Credit Author Contribution Statement

Humayun Majid Khan: Conceptualization, Methodology, Software, Validation, Writing – Original draft. **Samana Jafri:** Project Administration, Supervision. **Ansari Zoha Najmul Kalam:** Data Curation, Investigation, Writing – Review & editing. **Ansari Adnan Mohd Irfan:** Formal Analysis, Project Administration, Supervision. **Ashmaan Balan Shaikh:** Formal Analysis, Methodology, Writing – Review & editing.

Funding Declaration

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data Availability Statement

The data supporting the findings of this study are available from the corresponding author upon reasonable request. The source code developed for the prototype is not publicly available.

Conflict of Interest

There is no conflict of interest.

Artificial Intelligence (AI) Use Disclosure

The authors declare that artificial intelligence (AI)-assisted tools were used only for language refinement, grammar improvement, and manuscript structuring purposes during the preparation of this work. All technical content, experimental implementation, results,

and interpretations were independently developed and verified by the authors.

Supporting Information

Not applicable

References

- [1] T. Mackey, G. Nayyar, A review of existing and emerging digital technologies to combat the global trade in fake medicines, *Expert Opinion on Drug Safety*, 2017, **16**, 587–602, doi: 10.1080/14740338.2017.1313227.
- [2] K. Clauson, E. A. Breeden, C. Davidson, T. K. Mackey, Leveraging blockchain technology to enhance supply chain management in healthcare, *Blockchain in Healthcare Today*, 2018, **1**, doi: 10.30953/bhty.v1.20.
- [3] World Health Organization (WHO), A study on the public health and socioeconomic impact of substandard and falsified medical products, Geneva, Switzerland, 2017, Available at: <https://www.who.int/publications/i/item/9789241513432>.
- [4] J. A. DiMasi, H. G. Grabowski, R. W. Hansen, Innovation in the pharmaceutical industry: New estimates of R&D costs, *Journal of Health Economics*, 2016, **47**, 20–33, doi: 10.1016/j.jhealeco.2016.01.012.
- [5] F. Casino, T. K. Dasaklis, C. Patsakis, A systematic literature review of blockchain-based applications: Current status, classification and open issues, *Telematics and Informatics*, 2019, **36**, 55–81, doi: 10.1016/j.tele.2018.11.006.
- [6] H. M. N. D. Bandara, S. B. X. Eberz, R. M. N. M. Rathnayake, Blockchain technology in the pharmaceutical industry: A systematic review, *PeerJ Computer Science*, 2022, **8**, e840, doi: 10.7717/peerj-cs.840.
- [7] K. Christidis, M. Devetsikiotis, Blockchains and smart contracts for the Internet of Things, *IEEE Access*, 2016, **4**, 2292–2303, doi: 10.1109/ACCESS.2016.2566339.
- [8] U. Bodkhe, S. Tanwar, K. Parekh, P. Khanpara, S. Tyagi, N. Kumar, M. Alazab, Blockchain for industry 4.0: A comprehensive review, *IEEE Access*, 2020, **8**, 79764–79800, doi: 10.1109/ACCESS.2020.2988579.
- [9] A. Azaria, A. Ekblaw, T. Vieira, A. Lippman, MedRec: Using Blockchain for Medical Data Access and Permission Management, 2nd International Conference on Open and Big Data (OBD 2016), IEEE Computer Society, 2016, 25–30, doi: 10.1109/OBD.2016.11.
- [10] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, *Decentralized Business Review*, 2008, Available at: <https://bitcoin.org/bitcoin.pdf>, Accessed 17 September 2026.
- [11] V. Buterin, A next-generation smart contract and decentralized application platform, *Ethereum*

- White Paper*, 2014, Available at: <https://ethereum.org/whitepaper/>.
- [12] C. Agbo, Q. Mahmoud, J. Eklund, Blockchain technology in healthcare: A systematic review, *Healthcare*, 2019, **7**, 56, doi: 10.3390/healthcare7020056.
- [13] E. Weyl, P. Ohlhaver, V. Buterin, Decentralized society: Finding Web3's soul, Available at SSRN 4105763, 2022, doi: 10.2139/ssrn.4105763.
- [14] T. Chen, X. Li, X. Luo, X. Zhang, Under-optimized smart contracts devour your money, 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2017, 442–446, doi: 10.1109/SANER.2017.7884650.
- [15] R. C. Merkle, A digital signature based on a conventional encryption function, *Advances in Cryptology—CRYPTO '87*, Springer, 1988, **293**, 369–378, doi: 10.1007/3-540-48184-2_32.

Publisher Note: The views, statements, and data in all publications solely belong to the authors and contributors. GR Scholastic is not responsible for any injury resulting from the ideas, methods, or products mentioned. GR Scholastic remains neutral regarding jurisdictional claims in published maps and institutional affiliations.