# Blockchain-Based Supply Chain Transparency

Web Realm Explorers
*Jamia Millia Islamia*

## Contents

## 1 Introduction

### 1.1 Problem Description

A supply chain consists of the series of activities and organisations that materials move through on their journey from initial suppliers to final customers.[2] Here, each node is prone to fraud and corruption to the detriment both farmers and consumers. The solution in a traditional supply chain is to introduce a trusted central authority to manage this. However, this does not solve the fundamental problem and only shifts the problem from trusting the nodes to trusting the central authority.

### 1.2 Decentralized Solution

The ethereum blockchain is a widely used transaction-based state machine.[3] This means that each transaction in the blockchain can allow it to map to a new state. This is acheived by small executable pieces of code known as smart contracts. We propose a solution in the ethereum blockchain to this problem by using smart contracts which will be decentralized, transparent, trustless and secure.

## 2 Ethereum and Smart Contracts

**Smart Contracts are immutable pieces of code executed on the Ethereum VM**

### 2.1 World State

The world state in ethereum is a mapping from 160-bit addresses to an account state.[3] Each account represents either a user's account or a smart contract. The account state $\sigma[a]$ consists of the following:

- **nonce**: The number of transactions sent from this address.
- **balance**: The balance of the account.
- **storageRoot**: A 256-bit hash of the Merkle-Patricia tree.
- **codeHash**: The hash of the code in this account.

$$\sigma : \{a \in \mathbb{N} | 0 \leq a < 2^{160}\} \rightarrow \texttt{AccountState}$$

### 2.2 Transaction

A transaction $T$ in ethereum is a cryptographically signed instruction by an actor. This transforms the world state into a new world state according to a state transition function $\Upsilon$

$$\sigma' = \Upsilon(\sigma, T)$$

### 2.3 Smart Contracts

A Smart Contract is an immutable piece of code that's stored in an Ethereum account which can be called by other actors to transform the world state. Each ethereum node executes the smart contract in a stack-based virtual machine (EVM). There is a fee associated with execution of each instruction in the contract and the use of storage, often called gas fees. Because of its immutability it is critical for a smart contract to be correct before being deployed and because of the associated gas fees it should also optimize as much storage since the price scales approximately quadratically.

## 3 Supply Chain Smart Contract

**A trustless and secure contract for transactions**

### 3.1 Description

We will denote the node that is paying in a supply chain in a transaction as buyer and the node that is selling as seller. This smart contract allows the buyer and seller to securely do a transaction in a decentralized system. The buyer initiates a transaction by sending a payment to the smart contract, coordinates and the maximum duration in which the item is to be delivered. If the seller delivers the item in the expected duration the buyer sends an acknowledgement causing the money to transfer from the smart contract to the seller. If the seller fails to deliver, the buyer has the option to cancel the transaction causing the money to return to the buyer. This contract code can't be tampered with due to the immutability of the blockchain.

### 3.2 Reputation System

We have solved the problem of a malicious seller not sending the item after getting the money, however, a malicious buyer can still refuse to send acknowledgement correctly. To mitigate this we introduce a reputation system, for all successful transactions both the buyer and seller gain 1 reputation point. For unsuccessful transactions both the buyer and seller are given a vote which they can use to upvote / downvote each other.

## 4 Security Considerations

**Methods to ensure correctness of the Smart Contract**

### 4.1 Security Problems

Immutability of smart contract is a double-edged sword, while it guarantees no one tamper with the smart contract code, it also means that the developer is not able to correct any security vulnerabilities post-deployment. To deal with this there must be ways to ensure that adequete techniques are used to gain confidence in the correctness of the smart contract.

### 4.2 Testing

#### 4.2.1 Unit Testing

We have written unit tests in solidity to check for the requirements of the functions.

#### 4.2.2 Fuzz Testing

We have also written fuzz tests in solidity which test the requirements by random input. This can cover for edge cases which a programmer may not notice when writing a unit test.

#### 4.2.3 Integration Testing

We have used `viem` and `node:test` in order to write integration tests that test multiple functions together. They can help detect bugs that arise from the interaction between multiple functions.

### 4.3 Static Analysis

We have used static analyzers such as `slither` to verify certain classes of bugs in compile time. These tools can greatly help reduce the time spent in debugging the code.

### 4.4 SMT Checkers

A Satisfiabile Modulo Theory (SMT) Checker are a tool that can mathematically prove the invariants of a program hold true. The solidity compiler can use SMT Checkers[1] such as `z3` in order to find any invariants can't be proven.

## 5 User Interface

**A friendly UI to allow anyone to use our DApp**

### 5.1 Tracking

We provide an interface for tracking the entire journey of an item from the farmer to all the distributors. This can be done by scanning a QR code of the item identifier which will provide a graph showing all the locations the item has been in.

### 5.2 Acknowledgement

The acknowledgement is also done by simply scanning the item identifier QR code to send the payment from the smart contract to the seller.

## 6 Conclusion

**We have build a decentralized, trustless and transparent Suppy Chain**

We have proposed and prototyped a decentralized, trustless and transparent applicaation for an agriculture Supply Chain. This system is implemented on the widely used Ethereum blockchain through the use of smart contracts. It has several security guarantees due to the use of multiple methods of verification such as testing, static analysis and smt checking. It provides a user-friendly interface to its users to use this system without any prior technical knowledge.

## References

[1] Solidity. *SMTChecker and Formal Verification.* URL: https://docs.soliditylang.org/en/latest/smtchecker.html.

[2] D. Waters. *Logistics: An introduction to supply chain management.* Palgrave Macmillan, 2003.

[3] Dr. Gavin Wood. “Ethereum: A Secure Decentralized Generalised Transaction Ledger”. In: (2025). URL: https://ethereum.github.io/yellowpaper/paper.pdf.