DLTs for Traceability
IOTA smart contract

Pozzoli Davide

GitHub: https://github.com/treyvian/BlockchainRetailSupplyChain

Project report for
Blockchain and Cryptocurrencies

Artificial Intelligence
University of Bologna
Italy

Contents

Chapter 1

Abstract

This paper represents the report for the work for the exam of Blockchain and Cryptocurrencies for the project proposal #10: DLTs for Traceability and create a smart contract that would be able to trace the goods in a supply chain. The deployment is done on the IOTA 2.0 testnet, available at the moment of writing. The idea behind the implementation comes from the Colnago announcement, where products with a unique identifiers can be tracked across the supply chain using blockchain technology. The smart contract present in the linked repository, tries to provide a possible solution in a very simplified manner, allowing to track a product from the manufacturer to the client and being able at every step to verify its origin.

Chapter 2

iota 1.0

IOTA is a scalable open source communication protocol with tokens (cryptocurrency) used for value transfers. It is developed and provided by the non-profit IOTA Foundation.

IOTA does not work according to the traditional blockchain technology, but works with the concept of the Tangle, which is a Directed Acyclic Graph – DAG.

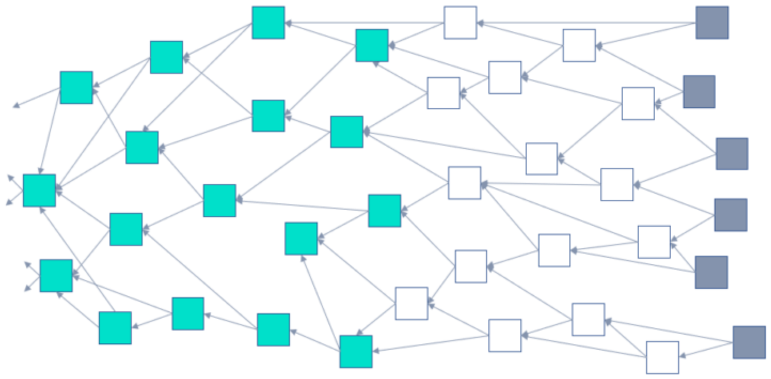
IOTA architecture includes the following basic components:

- *Clients*: Users of an IOTA network (wallets, apps, etc.) that send transactions to nodes to attach to the Tangle.
- *Nodes*: Connected devices responsible for ensuring the integrity of the Tangle. These devices form an IOTA network.
- *Tangle*: An attached data structure (public ledger, main ledger), which is replicated on all nodes in an IOTA network. All data in the Tangle is stored in objects called transactions. Instead of blocks that hold multiple transactions, each transaction is another node in the graph. When a transaction is attached to the Tangle, it cannot be changed and is immutable.

Transactions are organized in bundles because transferring IOTAs from one address to another requires several different transactions:

- An output transaction that increments the recipient's balance by the desired amount
- 1-3 inputs that authorise the spending of the IOTAs from the sender's address(es)
- (if necessary) a change transaction that sends any remaining amount to a new address owned by the sender

Bundles are atomic transfers either all transactions inside the bundle will be accepted or none.



In a highly simplified manner, the transaction process proceeds in five steps:

- *Create transaction*: With recipient address, sender, value and optionally with a message.
- *Sign*: By means of the private key (seed) the transaction is signed. This confirms that one is the owner of the entries made.
- *Tip selection*: A node selects two randomly unconfirmed transactions (tips) to be referenced in the new transaction using the (RWMC) Random Walk Monte Carlo algorithm. The selection of the tips is random but with a bias based on the cumulative weight to avoid lazy tips.
- *Proof of Work*: This is only needed for spam protection and not for consensus with a small arithmetic problem (nonce) to be solved. In addition, the tips assigned to you are checked to see if the account is sufficiently covered or if there are conflicting transactions.
- *Execution*: The finished transaction is finally sent to a node, which then distributes it in the network to all its neighbors (other nodes).

In IOTA there is no difference between miners and users. IOTA nodes only perform basic operations that do not require much processing power (e.g. storing the ledger, validating transactions). This removes the dichotomy between transaction makers and transaction authenticators. Basically anyone adding a transaction contributes also through ”mining”.

In the current IOTA implementation, nodes only trust (value) transactions that are referenced and approved by milestones issued by the coordinator, where the milestones are zero value transactions which full nodes refers to. The Open Source Coordinator can only validate transactions. It cannot bypass consensus rules, so it cannot create, freeze or steal tokens. For monitoring, this rule and the address of the coordinator is hard-coded in every node. The coordinator’s influence on the tangle is therefore very limited, because the tangle is additionally monitored constantly by all other nodes.

Chapter 3

iota 1.5 – Chrysalis

The objective of the IOTA Foundation was to optimize the IOTA mainnet before the release of IOTA 2.0 (Coordicide) to offer an enterprise-ready solution for the ecosystem. This was achieved by an intermediate update called Chrysalis.

3.1 Support for a new signature scheme

The IOTA 1.0 protocol was based on the Winternitz One-Time Signature (W-OTS) scheme, a hash-based signature scheme that uses the ternary 243-trit hash function Kerl. It has been shown to be resistant to a sufficiently powerful quantum computer, however unlike conventional signatures, it also has significant drawbacks, the signatures generated are quite large, which affect storage and speed.

Chrysalis introduces the new common signature scheme Ed25519, which completely replaces the old one. The Ed25519 is a modern signature scheme that uses SHA-512 and Curve25519. It drastically reduces the transaction size and consequently allows a significant increase in possible messages per second (mps). Another advantage is the re-usability of addresses, which greatly simplifies the implementation of IOTA technology. One disadvantage is that it is less quantum robust, but this is a problem that most conventional encryption systems currently have.

3.2 Atomic transactions

IOTA 1.0 uses the concept of bundles to create transfers. A disadvantage of bundles is mainly faced by developers, because it is much more complicated to get all transactions from a bundle and arrange them properly, instead of just processing a single message.

With the update to IOTA 1.5, the old bundle construct was removed, and the simpler Atomic transactions were introduced instead. Atomic transactions are much faster, more flexible (variable transaction size) and put less load on the network. They are also better suited for later Sharding / slicing than bundles.

3.3 Improved tip selection

3.3.1 White-flag Approach

The white flag confirmation algorithm which is a new approach to calculating credits is a simpler, conflict-avoiding approach that improves the speed and efficiency of tip selection, eliminates certain attacks (e.g., conflict spam), and significantly reduces the need for re-attachments. It allows milestones to confirm conflicting bundles by enforcing deterministic ordering of the Tangle and applying only the first bundle(s) that does not violate the ledger state.

It brings also some drawbacks: the ledger state is only well-defined at milestones. This means, we have to wait till each milestone is issued in order to confirm a spend (exactly the current mainnet status). To prove that a certain (non-milestone) transaction is valid, it is no longer sufficient to just provide the “path” to its confirming milestone, but, instead, all transactions in its past cone.

3.3.2 Tip selection

In IOTA 1.0 the algorithm used for tip selection was the “random walk”. This was essential for consensus building but also introduced some undesirable properties:

- Honest transactions could go neglected if they did not have enough weighting. This led to an increased need for promotions and re-attachments.
- Calculating the cumulative weights of transactions is relatively expensive and poses a problem for the scalability of the protocol.

Due to the white flag confirmation algorithm, it is no longer necessary to perform complex tip selection such as the “random walk”. Therefore, a simpler, more powerful *Uniform Random Tip Selection* algorithm can be used, which in turn increases overall message throughput.

To attach a new transaction to the Tangle, the algorithm can select and approve up to eight previous transactions – preferably tips. This approval mechanism represents the “belief” in the Tangle: If transaction y approves transaction x, it means that y believes that transaction x is valid and that its entire history is also valid. This brings the following benefits: Faster conflict resolution, reducing the likelihood that a transaction will be inadvertently attached to the wrong part of the Tangle.

3.4 Switch to an internal binary representation of the trinary transaction

A switch to an internal binary representation of the trinary transaction. This allows to work on binary data for validation, IO, and other processing, without the current reliance on binary-ternary conversions as in the pre-Chrysalis node software. The switch to binary transactions further reduces transaction size, saving network bandwidth and processing time.

Chapter 4

iota 2.0 – Coordicide

Coordicide is the next new protocol that is coming to IOTA. As before mentioned *Chrysalis* was a middle step to prepare for the final removal of the coordinator. The update is not live yet but the development is in progress in a testnet called *Nectar*. Chrysalis is secure and feeless, but with IOTA 2.0, it will be also fully decentralized, thanks to the removal of the Coordinator.

4.1 node identities and mana

The removal of the Coordinator alone is not sufficient for achieving decentralization. In fact, the consensus mechanism originally proposed in the IOTA white paper required that the majority of transactions always come from honest network participants. In other words, honest actors would need to own a majority of the network's hashing power. However, without miners, IOTA has no concept of constant, honest hashing power. The implication is that honest nodes would need to send a continuous stream of transactions, regardless of whether they are actually using the network.

At the core of the consensus mechanism there is a voting protocol through which nodes request the opinions of other nodes in order to decide which messages should be included in the Tangle, and which should be orphaned. Leading up to the current release, a number of challenges were solved to allow for the removal of the Coordinator.

Without the coordinator deciding which transactions to include, a valid message must have a consistent history without any conflicts. So when conflicts arrive, the Tangle forks into different branches. The IOTA 2.0 protocol must do two things:

- Decide which branches will survive.
- Record this decision for new nodes joining the network later.

Regardless of the complexity of the branches, the IOTA protocol always chooses a single branch, and then a single valid ledger state. To make these decisions, the IOTA 2.0 protocol works in a cycle:

Messages enter the network via the congestion control algorithm which manages access. Nodes vote on new conflicts through the voting protocol, **FPC**, which determines which branches should be rejected. FPC voting is protected from attackers by **mana**, a reputation system which fairly limits who can vote. After FPC rejects bad branches, tips are selected from the correct branches for new messages to reference. As these correct branches gain more messages, we say that their approval weight grows. Once a branch gets enough approval weight, its transactions are finalized and booked to the ledger, and mana is updated determining the next FPC voters.

- **Mana**: In each transaction, token holders pledge consensus mana to nodes. Since these pledges require tokens, attackers can only have so much mana.
- **FPC Voting**: When new messages enter the network, conflicts trigger the voting protocol Fast Probabilistic Consensus, or FPC for short. First, nodes decide which conflicting transactions they like based roughly on what arrived first. Then each node votes by directly asking a few other nodes which transactions they like. If enough of these nodes like a transaction (where "enough" is a random threshold), the node likes it too. Then each node again asks a few different nodes their opinion, repeating the process several rounds until all the opinions stabilize.
- **Mana protects FPC voting**: the more consensus mana a node has, the more often other nodes ask its opinion. Mathematical analysis and simulations show that FPC almost always terminates correctly with all honest nodes in agreement when the attacker has a significant amount of active consensus mana.
- **Tip Selection**: After FPC terminates, the protocol has effectively chosen which branches to reject: any branch containing a transaction disliked by FPC is rejected. When creating a new message, nodes randomly select tips (unreferenced messages) from the surviving branches.
- **Approval Weight**: The approval weight of a branch is essentially the amount of consensus mana held by nodes which issued a message on that branch. Since all honest nodes attach their new messages to non-rejected branches, the approval weight of these branches quickly becomes large, while the approval weight of a rejected branch remains small.

4.2 Ledger State

The introduction of a voting-based consensus requires a fast and easy way to both detect double spends and transactions that try to spend non-existing funds.

These conditions are fulfilled by the introduction of a Unspent Transaction Output (UTXO) model for record-keeping, which enables the validation of transactions in real time.

The UTXO model defines a ledger state where balances are not directly associated with addresses but with the outputs of transactions. In this model, transactions specify the outputs of previous transactions as inputs, which are consumed in order to create new outputs. This allows users to specify which funds are to be spent even if multiple parties have sent funds to the same address.

4.3 New Message Layout

The basic unit of information in the IOTA 2.0 protocol is an object called message which are gossiped through the network containing data and transactions. Together, messages form the tangle, central data structure of the IOTA protocol. Each message has several parts, including a header, payload, and signature.

The header of the message contains the essential information required for a node to process a message, like version number, parents nodes, timestamp, nonce, etc... Each message is concluded with a signature of the node whose ID is listed in the header. The signature signs the entire message, including the payload, which makes the contents of the message unalterable.

4.4 Timestamps

Every DLT has the need of a system that links information to time. In the legacy implementation and Chrysalis, the main time references are milestones: The milestone index of a message is the first milestone referencing it. With Nectar, decentralization brought the need for a new way to define those references: Timestamps. Timestamps are values declared and signed by the node issuing the message, and are expected to represent the issuance time of the message (from the point of view of the issuer node).

4.5 GoShimmer node

GoShimmer is a prototype node software in progress that allows nodes to reach consensus without the coordinator so that IOTA networks can be decentralized. The code in GoShimmer is modular, with each module being one of the Coordicide components or a basic node function. This approach allows to develop each module in parallel and test GoShimmer with one or more different versions.

The master branch is the stable version of the GoShimmer software, which includes a minimal set of modules that allow you to send zero-value transactions. The master branch includes the following coordicide modules node-Identities and autopeering.

Since everything is still under development many information we have about the features of the node may be different from the final release.

Chapter 5

IOTA Smart Contracts

The IOTA Smart Contract Protocol (ISCP) use the logic of UTXO Digital Assets (also known as Colored Coins). To create an SC, a new Digital Asset must be created and sent to the address of the SC. This transaction becomes the origin transaction (Genesis) of the SC. No additional fees or similar are charged for this process. This created Digital Asset, which represents the SC, remains at the address of the SC for the duration of the SC's term and is therefore the property of the SC.

Charges are necessary. They are an incentive to induce nodes to execute the SC code. With this solution, the whole network is not forced to execute every SC code (as with ETH). Instead, only a relatively small group of nodes is specially selected to execute a certain SC and, if necessary, rewarded accordingly. However, if there are other incentives to execute the SC code a fee is not required.

The IOTA Smart Contracts Fee Model:

- The fee model is hard-coded into the core protocol
- Preset: Fees = 0 (feeless)
- Fees = Validator fee + Chain owner fee
- The fee model can be extended by smart contracts for each use case

5.1 How do SCs work at IOTA?

IOTA SCs are defined as immutable state machines:

- **State Machine**: Every Smart Contract has a state that is connected to the Tangle. The state contains data like account balances, input conditions and consequences over time. Each state update represents a state transition on the tangle.

- **Unchangeable**: The Smart Contract's state and program code are both unchangeable because they are stored on the tangle. The state can be updated incrementally by appending new transactions to the tangle.

Although Ethereum's "on-chain" Smart Contracts are very popular because of their features, they have some significant drawbacks. The most prominent is that each node must keep a copy of the program code and status of the contract for each SC that exists. Each node in the network must execute exactly the same code when the SC is triggered. There is no limit to the number of nodes that must execute this identical code just to produce a single result. And the larger the network gets, the more processing is needed to achieve the same result. This is a huge barrier to scalability. In addition to the transaction fees you have to pay to be considered for inclusion in the ledger. You also have to pay gas fees to keep the program running long enough for it to be completed. This means that the cost of running these SCs becomes prohibitively high, except in certain use cases where the cost is relatively insignificant.

To facilitate use cases, including the Internet of Things, IOTA builds its SCs on layer 2, "off-Tangle".

As a result, IOTA SCs provide a more natural way to perform distributed computations. Each smart contract can be executed in a localized context without forcing the entire network to perform the same task. This also means that IOTA SCs will not become a barrier to scaling for the IOTA network with sharding solutions in the future.

5.2 Execution

The execution of an SC is performed by a committee of nodes of fixed size (with distributed sharing of private keys), which execute a program by consensus and send the results to the tangle. The fixed size is determined by the owner (issuer and operator) of the SC. For threshold cryptography hundreds are still a practical size. The minimum is one node, which means that it is only one centralized script that processes SC requests. Committee nodes are identified by their addresses in the network, known only to the owner, other committee nodes and other trusted (so called access-) nodes

5.3 Owner of a Smart Contract

Every smart contract has an owner who is responsible for:

- Creating the SC program and submitting it to the network.
- Determining the size of the committee (the number n) and selecting the nodes that will be part of the committee.
- Deciding how many committee nodes need to reach consensus on SC status updates. This number is called a quorum.
- Determining other general configuration parameters of the SC.

An owner may be a single entity, e.g., an organization or an individual, or it may be a decentralized collection of peers, e.g., a consortium of organizations. In either case, the owner only manages the establishment and configuration of the SC and does not participate in its operation.

5.4 The committee and the quorum

The account address of the smart contract is owned by the committee. This address contains the deposited IOTA tokens. Only a quorum of committee nodes is able to transfer tokens away from the address. In other words, a quorum of committee nodes must work together to transfer the IOTA tokens from the account.

IOTA uses the Boneh-Lynn-Shacham threshold signature scheme (BLS) for multisignature addresses to give the nodes on the committee equal ownership rights over the address. These addresses enable threshold signatures, which means that a certain number of nodes in the committee, the quorum, must sign the same transaction with their secret key.

Each node on the committee owns a secret share of the key and only collectively through the quorum consensus process (usually 2/3 plus 1), a valid signature can be generated, the state of the SC updated, and funds moved that are controlled by the SC.

5.5 How the status of a Smart Contract is updated

Committee nodes always listen to the Tangle for request transactions to be sent to the SC.

When a committee node detects a transaction, it computes the next state by running the SC program. In this way, honest and synchronized nodes will always generate exactly the same state update. Committee nodes never exchange real results, only hashes. So there is no risk of nodes copying each other's results to try to get the reward.

If nodes do not agree on the updated state, each will sign a different transaction, resulting in an invalid signature. Therefore, the nodes must reach a majority consensus on the updated state.

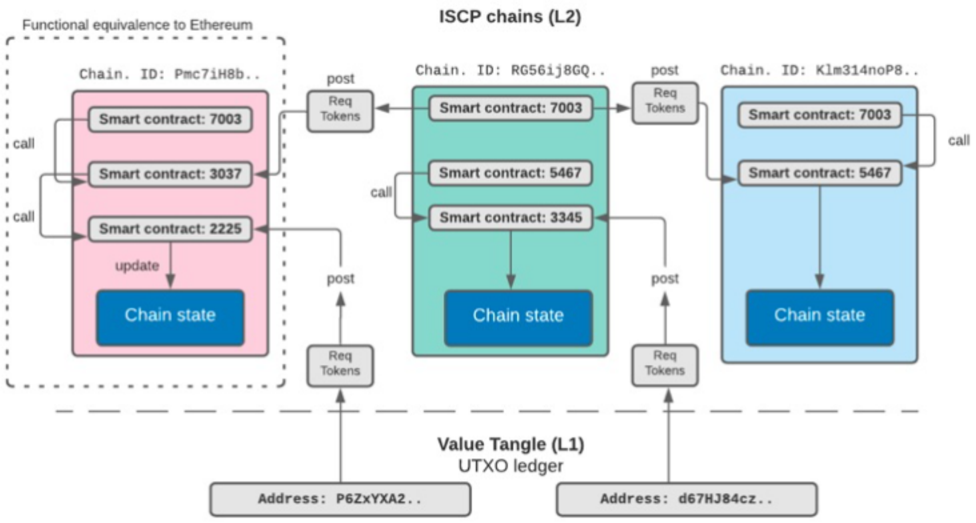
In the asynchronous setting of the Tangle, each committee node may see a different state depending on its local clock or its neighbors. Committee nodes must therefore run a distributed consensus algorithm to reach consensus on the outcome. This transaction must reference the previous state to create a chain of state updates that serves as an audit trail. When the transaction is committed, it becomes the new state of the smart contract.

5.6 Wasp: A Node for IOTA Smart Contracts

The IOTA Smart Contract Protocol (ISCP) is a 2-layer protocol built on top of the core protocol and executed by GoShimmer nodes. The development of this protocol is fully decoupled into a separate node called Wasp. Thus, IOTA's smart contracts are run through the network of Wasp nodes, all of which are connected to the Tangle.

Smart contracts are transacted through a so-called contract chain, the representation of the contract state. The Chain can contain many Smart Contracts, all working on the same global state of the Chain. From this perspective, the Contract Chain is essentially a blockchain anchored on the Tangle. IOTA Smart Contracts can be considered "classic" Smart Contracts, but with the added feature that you can have multiple such parallel Chains all using the same native IOTA token and trading between them in a trusted manner on the Tangle. This allows for trusted interoperability between different applications.

With the IOTA Smart Contracts Protocol, developers and enterprises are free to define their own flexible environments that meet their needs (Smart Contract languages / virtual machines) as well as sizes of validation committees that meet their required or desired level of decentralization and security. The IOTA Smart Contract Protocol allows them to run a "permissioned" Smart Contract chain validated by a committee of their own nodes, for example, or define a committee of nodes among consortium partners. ISCP is also built with the intention of running completely "permissionless", meaning that a committee of validators can be chosen from an open market of validators. All Smart Contract chains – whether open or private – benefit from the inherent security and interoperability that comes from anchoring each Smart Contract state and their outcomes on the IOTA feeless base layer.



IOTA Smart Contracts therefore do not require all nodes in the network to execute all Smart Contracts, but allow for a more flexible, meaningful definition that meets the needs of the Smart Contract owner. This will dramatically reduce cost and power while greatly increasing flexibility and not compromising on individual security requirements and the compatibility and interoperability required by dApps.

5.7 EVM

The current release of IOTA Smart Contracts has experimental support for EVM/Solidity smart contracts as well as Wasm based smart contracts. This means that existing smart contracts and tooling from other EVM based chains like Ethereum are fully compatible with EVM chains running on IOTA Smart Contracts. This allows us to offer the existing ecosystem around EVM/Solidity a familiar alternative.

With IOTA Smart Contracts, an EVM based chain runs inside an IOTA Smart Contracts chain as an IOTA Smart Contracts smart contract. Because of this, it is possible to run both Wasm based smart contracts and an EVM chain in a single IOTA Smart Contracts chain. They offer an EVM compatible JSON-RPC server as part of the wasp-cli, which allows you to connect to these EVM Chains using existing tooling like MetaMask, Remix or Hardhat. Deploying to a new EVM chain is as easy as pointing your tools to the address of your JSON-RPC gateway.

Limitations

There are some limitations you should be aware of at the time, which will be addressed in the months to come:

- The current implementation is fully sand-boxed and not aware of IOTA or IOTA Smart Contracts. It currently can not communicate with non-EVM smart contracts, nor can it interact with assets outside the EVM sandbox yet, like the Layer 1 of IOTA.
- You start an EVM chain with a new supply of EVM specific tokens assigned to a single address (the main token on the chain which is used for gas as well, comparable to ETH on the Ethereum network). These new tokens are in no way connected to IOTA, or any other token, but are specific for that chain for now.
- Because EVM runs inside an IOTA smart contract, any fees that need to be paid for that IOTA smart contract have to be taken into account while invoking a function on that contract. To support this right now the JSON-RPC gateway uses the wallet account connected to it.
- You need to manually deposit some IOTA to the chain you are using to be able to invoke these functions. We are planning to resolve this at a later phase in a more user-friendly way.

Overall these are temporary solutions, the next release of the IOTA Smart Contracts will see a lot of these improved or resolved.

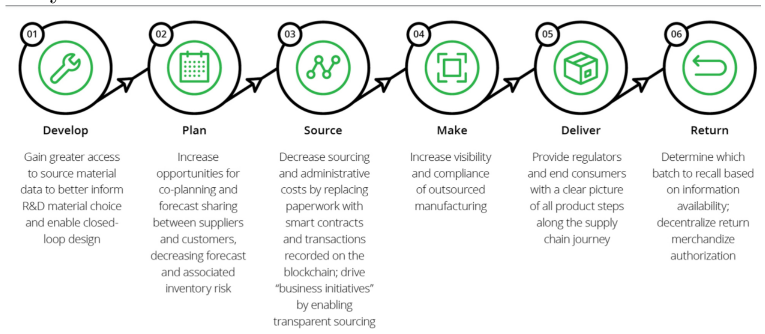
Chapter 6

My project

6.1 Supplychain with SC

In supply chain management the focus is on allowing a set number of known parties to conduct transactions with one another directly while improving security,ensuring contract compliance and reducing costs. Instead of coins, supply chain blockchains ”tokenize” a variety of transaction-related data, creating unique and readily verifiable identifiers for purchase orders, inventory units, bills of lading, etc.

Every participant in the chain has their own unique digital signature, which is used to ”sign” tokens moving through the chain. Every phase of a given transaction is recorded in the transfers between stakeholders, providing a built-in audit trail that can’t be tampered with since everyone gets their own copy of the chain. A bad actor tinkering with their own chain would also have to find a way to make the same changes to subsequent links in the copies maintained by everyone else.



01
Develop
Gain greater access to source material data to better inform R&D material choice and enable closed-loop design
02
Plan
Increase opportunities for co-planning and forecast sharing between suppliers and customers, decreasing forecast and associated inventory risk
03
Source
Decrease sourcing and administrative costs by replacing paperwork with smart contracts and transactions recorded on the blockchain; drive “business initiatives” by enabling transparent sourcing
04
Make
Increase visibility and compliance of outsourced manufacturing
05
Deliver
Provide regulators and end consumers with a clear picture of all product steps along the supply chain journey
06
Return
Determine which batch to recall based on information availability; decentralize return merchandize authorization

Companies using blockchain technology can expect to see several key benefits, including:

- Improved Efficiency: Because it relies on a shared network infrastructure, a supply chain using blockchain technology improves communication and collaboration for all parties. Greater traceability and transparency eliminate waste, duplicate orders and accounts payable headaches such as invoice fraud and rogue spend.

- More Ethical, Sustainable Sourcing: The traceability and tamper-resistance of the blockchain make it easier to verify where materials and goods come from.

- Additional Functionality For Other Digital Transformation Technologies: The blockchain readily integrates other technologies such as process automation and Internet of Things (IoT) objects such as smart sensors and RFID tags to further improve efficiency, visibility and accuracy throughout the value chain.

6.2 Implementation

The creation of a smart contract for managing a very compact supply chain. All the code can be found inside the repository linked at the beginning of this paper.

The smart contract is composed of a few files all contained inside the folder *contracts*. The main contract, SupplyChain.sol, is the one that contains all the main functions and the one that needs to be deployed. All the other contracts which are inherited by SupplyChain.sol are to define *Roles*. Roles are used to implement access control, which means that some functions can be called by an account only if it has the correct role. In my application roles are fundamentals since not everyone should have the possibility to add a new product on the market or being able to delete it.

The smart contract implemented uses 3 roles:

- Owner of the contract: it is the which execute the transaction to deploy the contract. It is basically a super-user, it can add Roles to other accounts can remove product from memory and execute every action available in the front end of the application. It is implemented in the file *Ownable.sol*

- Manufacturer : It has the ability to create new product and store them in memory. It is implemented in the ManufacturerRole.sol

- Retailer : It can buy products directly from the Manufacturer and resell them to the general public.

The contract is deployed on the IOTA testnet using Remix which is officially supported as specified in the wiki.

Bibliography

- https://v2.iota.org/
- https://wiki.iota.org
- https://govern.iota.org/t/conflict-white-flag-mitigate-conflict-spamming-by-ignoring-conflicts/233
- https://medium.com/51nodes/exploring-iota-2-0-smart-contracts-in-a-private-network-developing-a-prediction-market-c2d81988f75e
- https://www.forbes.com/sites/forbestechcouncil/2021/11/08/blockchain-in-supply-chain/?sh=2bfb71f54e1a