Logistics & Supply Chain Management Using the Power of Blockchain

Ghrushnesh Rathod
Department of Computer Science
Ramrao Adik Institute of Technology
Mumbai, India
email: ghr.rat.rt18@rait.ac.in

Shubham More
Department of Computer Science
Ramrao Adik Institute of Technology
Mumbai, India
email: shu.mor.rt18@rait.ac.in

Amit Patil
Department of Computer Science
Ramrao Adik Institute of Technology
Mumbai, India
email: ami.pat.rt18@rait.ac.in

Sheetal Ahir
Department of Computer Science
Ramrao Adik Institute of Technology
Mumbai, India
email: sheetal.ahir@rait.ac.in

Abstract: A large portion of supply chain networks have experienced hardships when attempting to incorporate all parts and partners, which make supply chain management network the board framework experiencing an absence of effectiveness and straightforwardness that make a consistent expansion in product misrepresentation and customer's mistake. All data about product creation, stockpiling and transportation should stream obviously during all phases of the production network by following and validating to keep away from item's tampering in the organization. Current software is based on top of brought together engineering and this leaves a hole for altering and bogus data particularly in metropolitan region, yet additionally the current arrangements can't permit the data to be imparted to authorities or buyers to successfully survey and guarantee recognizability and straightforwardness. This paper proposes use of blockchain to store details of product and logistics level to provide assurance to consumer regarding authenticity of product.

Index Terms: Blockchain, Ethereum, Smart Contracts, Decentralized Management

1. Introduction

Throughout the last many years, supply chain and logistics have significantly developed, excused and become a critical variable of seriousness and monetary accomplishment for practically all organizations. Supply chain has not exclusively been put at the focal point of market analysts' consideration, yet organizations spend an ever-increasing number of measures of their turnover on logistics and particularly transportation.

The worldwide exchange and store network, deluged by a developing intricacy of data and actual exchanges, face basic difficulties like consistence, discernibility, administration, higher exchange expenses, and security. In spite of the great money related valuation of the exchanges and the administrative investigation, the center information foundation is at present unfit to stay up with digitalization patterns.

Accomplishing greatness in logistics includes working cooperatively with others to advance the progression of actual products just as the complicated progression of data and monetary exchanges. However, today there is a lot of caught esteem in logistics, generally coming from the divided and serious nature of the logistics business. For instance, in the USA alone, it is assessed that there are more than 500,000 individual shipping organizations. With such countless partners associated with the supply chain, this frequently makes low straightforwardness, unstandardized processes, information storehouses and different degrees of innovation reception.

Many pieces of the logistics esteem tie are likewise bound to manual cycles commanded by administrative specialists. For instance, organizations should frequently depend on manual information section and paper-based documentation to hold fast to customs processes. This makes it hard to follow the provenance of merchandise and the situation with shipments as they move along the supply chain, causing grating in worldwide exchange.

Blockchain might conceivably assist with beating these grindings in logistics and acknowledge considerable increases in logistics process effectiveness. This innovation can likewise empower information straightforwardness and access among applicable supply chain partners, making a solitary wellspring of truth. Also, the trust that is needed between partners to share data is upgraded by the inherent security instruments of blockchain innovation. Besides, blockchain can accomplish cost investment funds by fueling more slender, more computerized, and mistake free cycles. Just as adding perceivability and consistency to logistics tasks, it can speed up the actual progression of products. Provenance following of merchandise can empower capable and manageable supply chains at scale and help to handle item falsifying. Also, blockchain-based arrangements offer potential for new logistics administrations and more creative plans of action. One of the innovations that emerged from this interest was blockchain, a DLT (Decentralized Ledger Technology) that was originally created in 2008 by Satoshi Nakamoto [1], and serves as the basis of Bitcoin and all other cryptocurrencies.

The rest of this paper is coordinated as follows. Section II presents the related work. Section III describes the proposed blockchain Ethereum solution implementation. Section IV gives the testing details and a discussion on the security and testing. Section V finishes up the paper and examines future work.

2. Existing System

Presently, the current frameworks / systems for overseeing records of production network and operations are centralized. A solitary database comprises of the multitude of information and following data of the item. Centralization alludes to an arrangement where whole data and information are put away at the highest point of the organizational structure. There are a couple of drawbacks of the concentrated framework referenced here. You want to enlist an accomplished and technical executive for keeping up with the database administration framework. We really want to purchase costly equipment, programming and generously compensated technical associates. At first, data set size isn't enormous size yet when we store a lot of information, then, at that point, the size of the database increments. Some particularly noteworthy academic studies on system proposals for blockchain and logistics have been done. Hasan and Salah [2] created an Ethereum application for a Proof of Delivery system. A major database can make a ton of issues. For instance, when the data of the database increases too much then the delay time of the result of the database queries can be increased.
Other disadvantages of centralized system in Supply chain use case:

- There are multiple handovers in the current system which leads to error in delivery time estimation.
- There are chances of wrong product delivery due to error in the system.
- A lot of paperwork is involved in the existing system.
- Sometimes, counterfeit products are delivered to the customer.
- Lack of proper authentication across different stages.
- Execution errors—such as mistakes in inventory data, missing shipments, and duplicate payments—are often impossible to detect in real time.

The authors in [3] proposed a simple scheme based on Ethereum blockchain which involves transporting a product between two parties. The scheme depends on a single key that is given to the transporter by the seller [3]. The key is transported along with the item and is handed over to the receiver who is the buyer. The buyer then needs to enter the key for verification. The key hash would already be available in the smart contract which acts as an escrow and holds the buyer's Ether. The Ether would only be placed in the seller's account if the hash of the key entered by the buyer matches the existing hash in the smart contract. A successful verification leads to the transfer of the Ether to the seller [3]. This solution is easy to implement as it is simple and depends on only one key. The method however, depends on trusting the transporter completely that no manipulation of the key would take place before reaching the buyer.

3. Technologies and Methodologies

3.1 Blockchain

The Blockchain is another innovation that tries to make data storage and transmission safer, straightforward, and decentralized. It's an appropriated information base, or public/private shared record of every advanced event, that has been run and shared by the blockchain's representatives. As far as innovation, Blockchain is a circulated framework that stores all exchanges as squares from its initiation. These are comprised of information that clients send in. Cryptographic calculations protect these information moves.
Blockchains can be summed up and used to build a bunch of settled upon decides that nobody can break, including clients and framework administrators. For multi-part applications requiring minimal common trust, they depend on a solitary framework compositional stage. The decentralization of information is a critical component of blockchain innovation, which likewise gives a more significant level of information assurance.
While the blockchain isn't the main type of appropriated record innovation, it is the most normal structure among digital currencies from which the modern applications talked about in this article are drawn. The conveyed design of the blockchain gives extra security looked at conventional "client server" models.
Blockchains can possibly assemble another age of transactional applications that set up trust, responsibility, and straightforwardness at their center while smoothing out business processes and legitimate requirements [4][5].

3.2 Application of Blockchain:

The mechanical insurgency, just as the expansion of information assortment parts, has raised worries about information security and correspondence dependability. The capacity, unwavering quality, recognizability, and realness of data gave by blockchain, just as insightful authoritative arrangements for a trust less climate, predict a key change in production network the executives. Moreover, pursuing the direction toward savvy coordinated operations and the utilization of correspondence and data innovation at all levels of the worth chain, the Blockchain offers critical potential for upgrading the strategies cycle. Quite possibly the most urgent part of a blockchain application is its connection with this present reality, which requires the utilization of pertinent apparatuses and advances.
Blockchain uses decentralized, conveyed record to work with the development of products, data and funds. It incorporates utilization of smart contracts to automate the supply chain capacities.
Blockchain Technology may not replace legacy systems or old applications soon. However, Blockchain can certainly be a complementary application to legacy systems and may even lead to the development of new systems in the near future [6].

3.3 Ethereum

The Ethereum Network gives an adaptable improvement environment. It is a P2P network that can interaction any sort of smart contract, which can be handily made with a couple of lines of code, and without the need of making your own unique reason blockchain foundation. Instead of Bitcoin and other single-reason blockchains, Ethereum decoupled this savvy contract layer, which presently runs on top of the fundamental Ethereum blockchain, making it simple to make smart contracts with only a couple of lines of code.
As indicated by Dr. Gavin Wood, prime supporter of Ethereum association, Ethereum, taken overall, can be considered a transaction-based state machine: we start with a beginning state and steadily execute exchanges to transform it into some last state. It is this last state which we acknowledge as the canonical "version" of the world of Ethereum [7]. It permits everybody to set their own guidelines in exchanges and state progress capacities. This is finished by including shrewd agreements, a bunch of cryptographic guidelines that are executed under the conditions that the maker of the agreement has set up.

3.4 Smart Contracts

The idea of “Smart Contracts” was presented by N. Szabo in 1994 in an unpublished composition and afterward officially in 1997 [8]. Smart Contracts are basically blockchain-based applications, briefly, self-executing programs which contain the terms and arrangements between the parties involved. A smart contract is addressed by its 20-byte address, like an EOA address, and it tends to be created in high level languages, like Solidity. At the point when the contract is conveyed on the organization, the code is assembled as bytecode. The component that separates Smart Contracts is that they consequently check whether or not the particulars of the understanding have been satisfied. Furthermore, to guarantee unwavering quality, adaptation to internal failure, and straightforwardness, the codes (Smart Contracts) are recreated on numerous hubs in the blockchain. There is an endless measure of contracts that can be made in the organization. Nonetheless, there are some "layouts" that are ordinarily utilized, as they give a few primary functionalities to gets that need to accomplish comparative practices. These are token norms that characterize a typical rundown of rules and are called ERC (Ethereum Request for Comments). Ethereum smart contracts empowered blockchain by allowing the execution of code [9].

4. Proposed blockchain solution

In this section we describe our Ethereum blockchain solution that makes use of security provided by this technology to create a solution between different levels of supply chain and logistics. Our solution can handle multiple entities at the same time. Our solution can be extended to multiple products and multiple levels of supply chain. Product details are also included in this solution which can be verified before the transaction.

4.1 System Overview

The proposed blockchain solution prioritizes proof-of-delivery of physical commodities being transferred across different entities in supply chain. Figure 1 represents multiple entities interacting with smart contract.

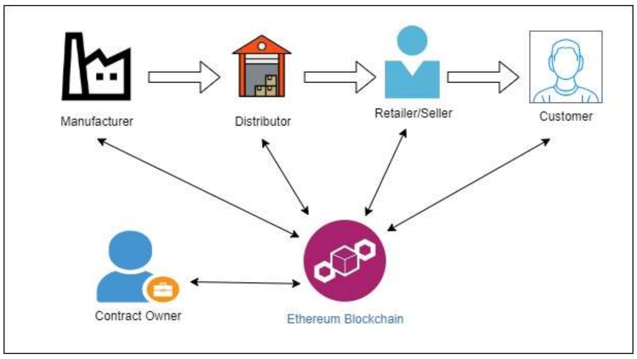


Fig. 1. Different Entities and their interaction with Smart Contracts stored in Blockchain

Each entity can interact with smart contracts stored in blockchain if they have required permissions. All the data of product is stored in smart contracts.
The roles of the participants can be summarized as follows:

- **Owner:** Owner is the one who deploys the contract to the blockchain network.
- **Manufacturer:** The manufacturer manufactures the product and add the details of product to the contract in the chain. He can put the product on sale.
- **Distributor:** The distributor buys the product from Manufacturer and gains ownership of the product to sale it to next level of people on supply chain.
- **Retailer/Seller:** Seller buys the product from distributor and then passes it to customer finally to complete the transaction process.
- **Customer:** Customer gains the ownership of the product and is free to pass on the product through resale as well.

Blockchain provides transparency and the ability to log the events on public ledger. Ethereum provides ability to store and fetch events which helps in tracking the product across the supply chain and helps to know the current owner of product easily. To provide verification of product across the supply chain product details are hashed and stored along with it. This helps in transfer of product across the chain easily and validation is provided. Blockchain is used to store details of products and transaction details between any two entities.
In order to fulfill these requirements of transparency and traceability of the product, smart contracts consist the following [10]:

- **Modifiers:** Modifiers are utilized in the smart contracts to make a prerequisite before the execution of a function. Modifiers were likewise used to confine the execution of a function in view of the Ethereum address of the function caller.

- **Methods:** Methods are used in Smart contracts to create function calls. Functions are responsible to perform specific set of actions. In this project methods are used to implement various functions to avoid repetition of code.
- **Events:** Events go about as notifications and are utilized as logs which can help in tracing back a product if there should be an occurrence of issues. Hence, any work that is executed makes an event that refreshes all entities about the status of product.
- **Variables:** Variables are used to store data that can be updated as they move across the chain. In this project variables are used to store product details and hash of the product.

4.2 System Design

The process of supply product from manufacturer and consumer consists of various entities. This contract is designed to store data of product across the entities and provide details of current and previous owners. To avoid the handling multiple contracts across different entities single contract was designed to store the entire information of supply chain. In multiple contracts are used to store data which makes the ecosystem confined to a particular product. Contracts are designed in such a way that all the data is stored in single contract and only the owner of product can transfer the product to next entity.

Once the manufacturer adds the details of product in contract the data is encrypted to create a hash will be used later during transfer of ownership between any two entities. During the process the buyer will have to verify the product details and hash of details to get the complete ownership. In this the system relies on buyer to make sure he verifies the details properly before making payments through ether. The general view and code structure of the smart contract was developed using Truffle [11], a testing tool for smart contracts, and Ganache-cli [11], an Ethereum blockchain simulator. Fig 2 gives clarity about the use of blockchain to store information of transaction and details of products in smart contracts.

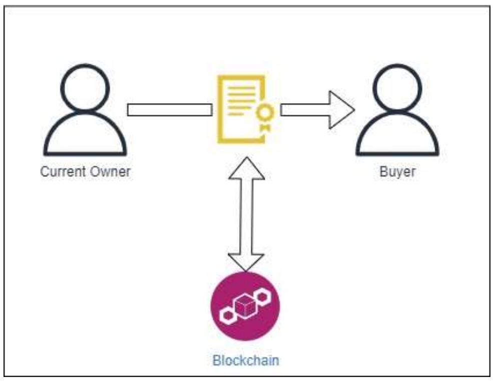


Fig 2: Transfer of ownership and transaction details are stored in blockchain

5. Implementation Details

In a Software, where blockchain is one of the parts, they may have to utilize the information put away on blockchain and the shrewd agreements running on blockchain to really take a look at specific conditions. ID of the exchanges or blocks on blockchain is a piece of information that can be effortlessly incorporated into the current information. Approval of the information can be executed by smart contracts running on blockchain.

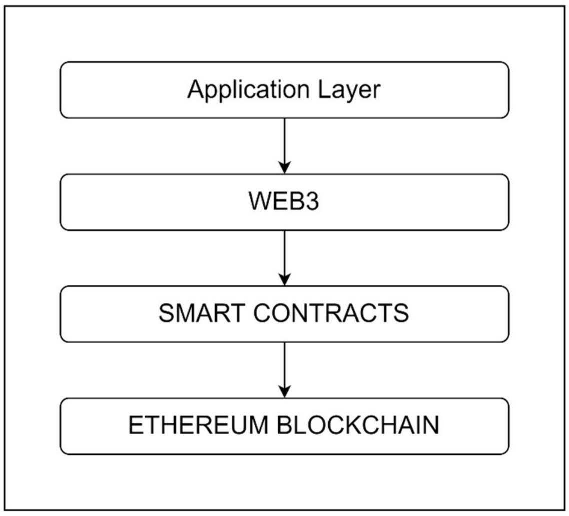


Fig. 3. General Implementation of System

Fig 3 gives information about overall architecture of the application. The system is divided into four layers: application layer, service layer, contract layer and data layer.

The application layer provides the user interface, and the service layer provides Node.js Web services needed by the system. The contract layer completes business logic control and access control of blockchain data and database data by smart contracts. The data layer is responsible for storage of blockchain data and database data. The invocational connection between service layer and contract layer is realized through RPC interface exposed by Ethereum.

Fig 4 gives information about the working and coordination between smart contracts across levels to generate a data layer for information to be stored in blockchain.

The main working routine of the distributed network is as follows:

The node for every substance produces new exchange information, and engenders the information through the shared organization.

Every one of the hubs create the new squares as per the Ethereum agreement instrument.

Every one of the hubs compute the mark and hash, and the principal that completes the estimation and broadcasts the outcome can get the option to record this exchange.

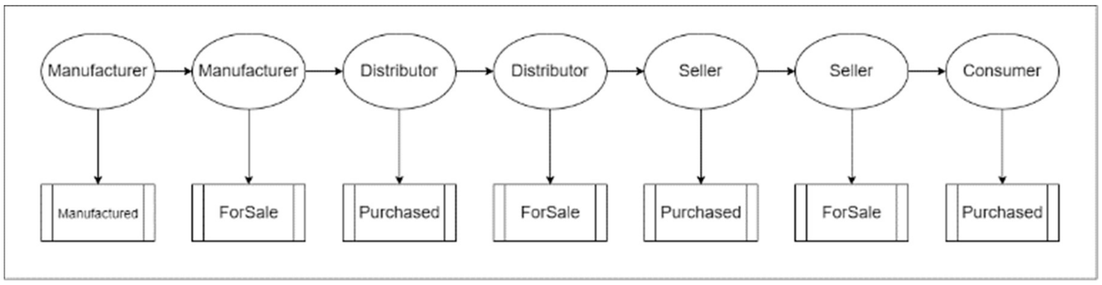
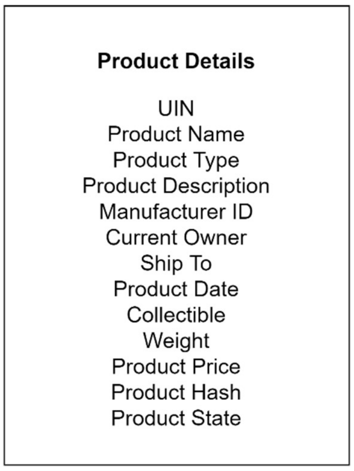


Fig. 4. Overall flow of a project

The exchange data, past hash, timestamp, difficulty and different fields would be stored with the new block;
The hub that gets the option to record the exchange communicates the new square to the whole organization.
The wide range of various hubs that get the new block will do the data check and put the new block to the tail of the blockchain.
The contracts are made and tried utilizing the Remix IDE which gives the fundamental devices to testing and investigating. Thus, making it simpler to change the code on a case-by-case basis while programming. This segment centers around the execution calculations utilized in the savvy contracts. The language utilized for composing the savvy contracts is Solidity. Three sorts of agreements are made as referenced in the Design segment which are the fundamental contract, CS gets that rely upon the quantity of carriers and the BT end of chain contract. One of the fundamentals viewpoints in the execution is that each agreement across the chain has the agreement locations of both the parent and the youngster contract.
Figure 5 gives information about the details to be stored regarding the product in the blockchain. Some set of details will be used for creating of product hash which will be constant throughout the supply chain. Refer figure 6 for flow of contract between 2 entities.


Product Details

UIN
Product Name
Product Type
Product Description
Manufacturer ID
Current Owner
Ship To
Product Date
Collectible
Weight
Product Price
Product Hash
Product State

Fig. 5. Item Details stored in Blockchain

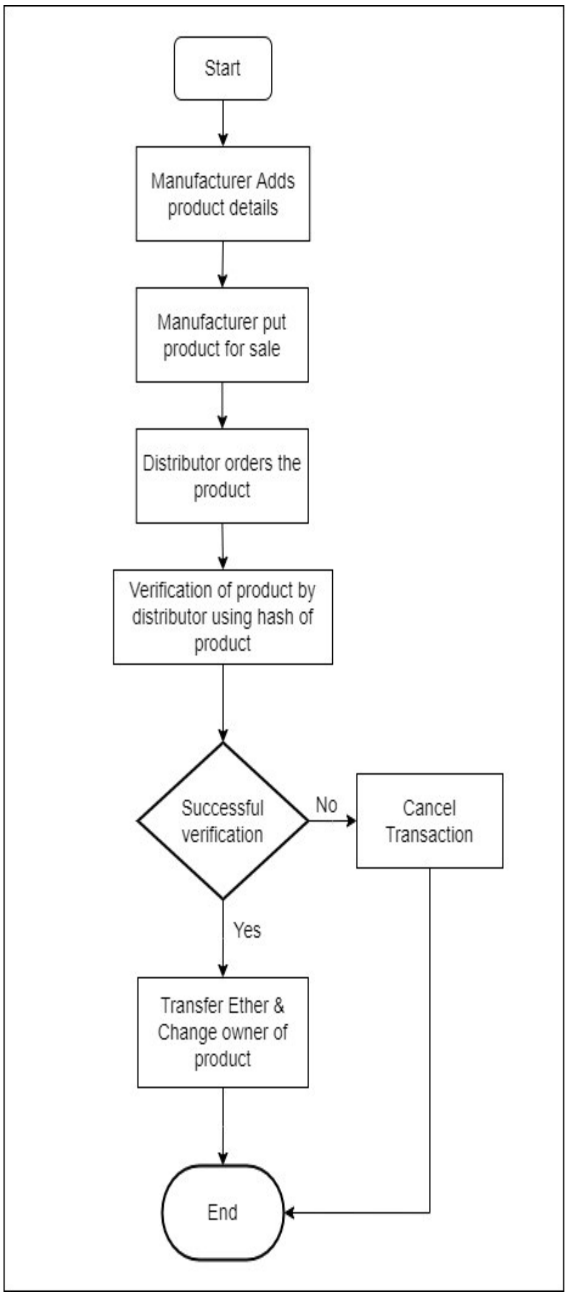

Start
Manufacturer Adds product details
Manufacturer put product for sale
Distributor orders the product
Verification of product by distributor using hash of product
Successful verification
No
Cancel Transaction
Yes
Transfer Ether & Change owner of product
End

Fig. 6. Shows transaction and transfer of ownership between 2 entities.

Algorithm 1: Create product by Manufacturer

Input: Product name, Description, Type, collectible, weight, manufacturer address

```
1. If(address belongs to manufacturer)
2.     uin = randMod() // to generate UIN from algorithm 2
3.     item = new Item(Product name, Description, Type, collectible, weight, manufacturer address) ;
4.     update product hash from algorithm mentioned in algorithm 3.
5.     Make manufacturer owner of product
6.     Add item to manufacturer products list
7.     Emit event indicating product creation in blockchain
8. Else
9.     Show error
```

5.1. Creating product by Manufacturer

Algorithm 1 shows how a product can be created by Manufacturer. For manufacturer to be able to add a new product in supply chain, it is mandatory he needs to have the required rights. Address of manufacturer can be added as an entity in Supply chain by the owner of main supply chain contract. Manufacturer needs to enter the product details to be able to create the product. Once the product is created, manufacturer is the owner of product by default. Events are used to indicate creation of product in blockchain.

Algorithm 2: Generate UIN of product

Input: address of sender, currentTimeStamp

```
1. randNonce = randNonce + 1
2. pack address of sender, currentTimeStamp and randNonce
3. Use keccak256() to convert into 256-bit hash.
4. Convert hash to uint
5. return remainder of uint by % 82589933
```

5.2. Keccak

Keccak is a family of sponge function instances, encompassing capacity values ranging from 0 to 1599 bits. All these instances are well-defined and so are their security claim. Keccak is the result of using the sponge construction on top of the Keccak-f permutations and applying the multi-rate padding to the input. Using multi-rate padding causes each member of the Keccak family (and in particular for each value of the capacity) to act as an independent function. As a native feature, Keccak provides variable output length, that is, the user can dynamically ask for as many output bits as desired [12]. Keccak is used for hashing purpose during verification in Algorithm 2, 5 and 6 mentioned here.

Algorithm 3: Sale by Manufacturer

Input: address, uin (product id), address of Receiver, price

```
1. if address is of manufacturer
2.     if product is owned by address
3.         update product visibility to true
4.         update product state to for Sale by Manufacturer
5.         update product price to price
6.         emit event on blockchain network
7.     else
8.         fail transaction
9. else
10.    fail transaction
```

5.3. Sale by Manufacturer

Once the product is manufactured by manufacturer it is up to him to decide the price and add the product for sale. The contracts first verify the address mentioned is of the manufacturer and is mapped with the current owner of the product. In case any of these conditions fail the transaction is cancelled. Once the conditions is satisfied the product visibility is changed to true and state is updated to "For Sale By Manufacturer". Default product price is updated to price decided by the manufacturer. Event is emitted regarding the product in blockchain.

Algorithm 4: Shipped by Manufacturer

Input: manufacturer address, distributor address, uin (product id)

```
1. if transaction address is manufacturer
2.     if receivers address is distributor
3.         if product state is sale by manufacturer
4.             change product state to "shipped by Manufacturer".
5.             change "Ship to Address" to distributors address
6.             update product hash using algorithm 5
7.             emit event of product to "Shipped by Manufacturer".
8.         Else
9.             error
10.    Else
11.        error
12. Else
13.    Error
```

5.4. Shipped by Manufacturer

Off the chain discussion regarding sale will take place between manufacturer and distributor which will be executed on blockchain. Address of entity who called the function will be verified if it belongs to manufacturer. The manufacturer then adds the address of the user he wants to ship the product to. The address will be verified if it belongs to distributor and will be processed for further steps. The product state is updated to "Shipped by Manufacturer". Using the steps mentioned in Algorithm 5 you can update the product hash which will be used for verification during algorithm 6. The event will be emitted to blockchain network.

Algorithm 5: Generate Product Hash using Keccak 256

Input: address1 (owner), address2 (receiver), uin (product id), productName

```
1. "pack" uin, address1, address2, product name using
   abi.encodePacked.
2. convert this packed data to hash using keccak 256
3. convert the obtained hash to integer.
```

5.5. Received Product by Distributor

In this part the distributor i.e., the receiver will be confirming the product transfer and will transfer the amount in the form of ether to previous owner of the product. Algorithm 6 carries out the execution of this part and transfers ownership of the product to distributor.

Algorithm 6: Received Product by Distributor

```
Input: uin, address1, address2, amount
1.  if productPrice <= amount
2.      if product state is shipped by manufacturer
3.          if ("product Hash" is equal to hash generated
            by algorithm 5 from "uin", "address1",
            "address2", "product Name")
4.              update product state as "Received by
                Distributor"
5.              update visibility of product to false
6.              transfer price of product to previous
                owner
7.              update current owner to distributor
                address
8.              add product to distributors product list
9.              emit an event "Received by distributor"
                to network
10.         else
11.             cancel transaction
12.     else
13.         error
14. else
15.     error
```

5.6. Hash Verification:

Hash verification occurs every time the product is transferred to next entity. Receiving entity verifies the hash value of the product details with the hash value stored in the contract to complete the transaction. Once the verification is completed ownership of product is transferred to receiver's address. Refer figure 7 for verification process.

By changing the entities and roles mentioned in the algorithm, the same algorithm is used by distributor to sell product to retailer and retailer to customer. Depending upon the type of commodity customer can also sell this product to another consumer with ease.

6. Testing and Validations

This section describes the testing and validation of smart contracts. Manufacturing of product by manufacturer and sale of the product to distributor will be tested in this section. Transaction will take place between manufacturer and distributor in this section. Multiple entities across the chain will be go through the process of transfer of product before reaching the customer.

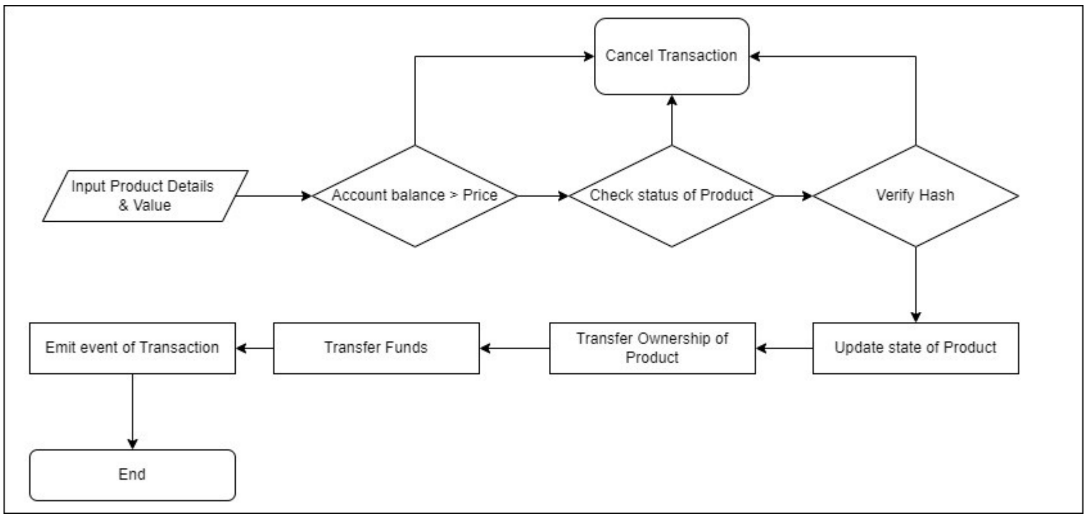
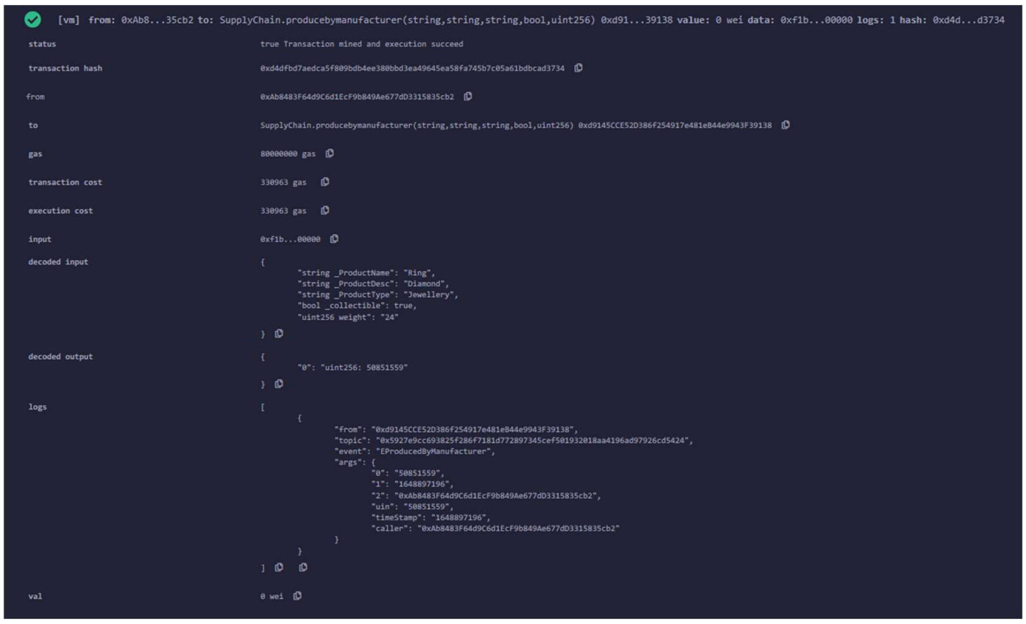


Fig. 7. Verification of Product and Receiver



```
[vm] from: 0xAb8...35cb2 to: SupplyChain.producebymanufacturer(string,string,string,bool,uint256) 0xd91...39138 value: 0 wei data: 0xf1b...00000 logs: 1 hash: 0xd4d...d3734
status                  true Transaction mined and execution succeed
transaction hash        0xd4dfbd7aedca5f809bdb4ee380bbd3ea49645ea58fa745b7c05a61bdbcad3734
from                    0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to                      SupplyChain.producebymanufacturer(string,string,string,bool,uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                     80000000 gas
transaction cost        330963 gas
execution cost          330963 gas
input                   0xf1b...00000
decoded input           {
                                "string _ProductName": "Ring",
                                "string _ProductDesc": "Diamond",
                                "string _ProductType": "Jewellery",
                                "bool _collectible": true,
                                "uint256 weight": "24"
                        }
decoded output          {
                                "0": "uint256: 50851559"
                        }
logs                    [
                                {
                                        "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
                                        "topic": "0x5927e9cc693825f286f7181d772897345cef501932018aa4196ad97926cd5424",
                                        "event": "EProducedByManufacturer",
                                        "args": {
                                                "0": "50851559",
                                                "1": "1648897196",
                                                "2": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                                                "uin": "50851559",
                                                "timeStamp": "1648897196",
                                                "caller": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"
                                        }
                                }
                        ]
val                     0 wei
```

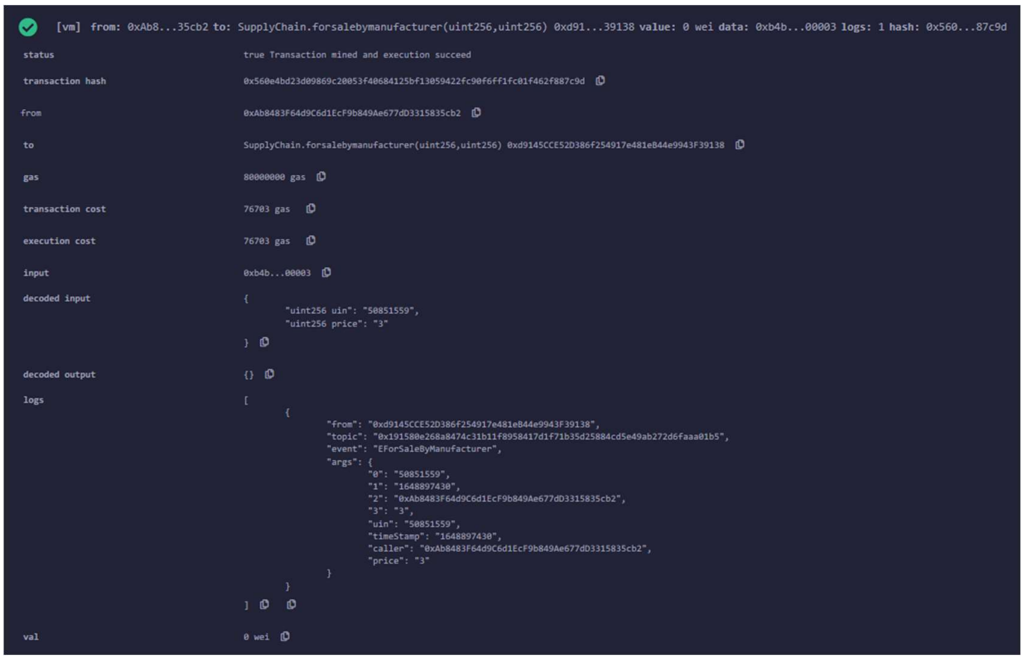
Fig. 8. Product is successfully added to the blockchain.

6.1. Add Product to Supply chain

Only the manufacturer can add the product to supply chain. Figure 8 shows that the product is successfully added to chain.
Once the product is successfully added to the chain the contract returns a unique identification number for the product which remains constant throughout the chain. Logs are generated which gives information about the transaction. Event is emitted in blockchain.

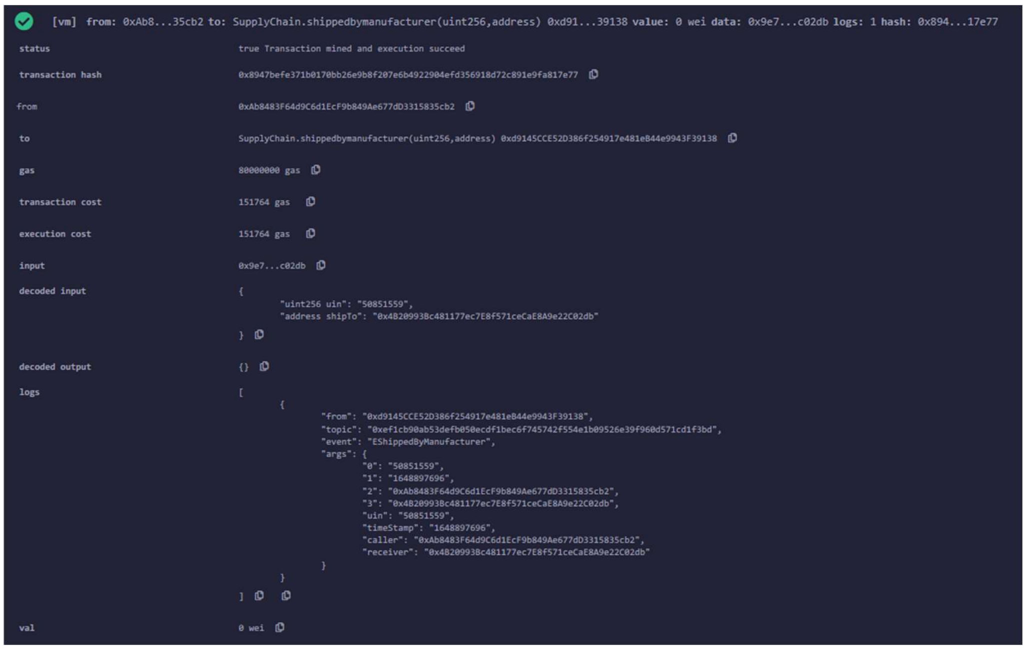
6.2. Sale product by Manufacturer:

Once the state of product is updated to "For sale by manufacturer" event in emitted in blockchain.
Manufacturer depending upon his convenience he can add the product to sell at the specific price. Figure 9 shows how the product can be updated for sale purpose.



```
[vm] from: 0xAb8...35cb2 to: SupplyChain.forsalebymanufacturer(uint256,uint256) 0xd91...39138 value: 0 wei data: 0xb4b...00003 logs: 1 hash: 0x560...87c9d
status                  true Transaction mined and execution succeed
transaction hash        0x560e4bd23d09869c20053f40684125bf13059422fc90f6ff1fc01f462f887c9d
from                    0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to                      SupplyChain.forsalebymanufacturer(uint256,uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                     80000000 gas
transaction cost        76703 gas
execution cost          76703 gas
input                   0xb4b...00003
decoded input           {
                                "uint256 uin": "50851559",
                                "uint256 price": "3"
                        }
decoded output          {}
logs                    [
                                {
                                        "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
                                        "topic": "0x191580e268a8474c31b11f8958417d1f71b35d25884cd5e49ab272d6faaa01b5",
                                        "event": "EForSaleByManufacturer",
                                        "args": {
                                                "0": "50851559",
                                                "1": "1648897430",
                                                "2": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                                                "3": "3",
                                                "uin": "50851559",
                                                "timeStamp": "1648897430",
                                                "caller": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                                                "price": "3"
                                        }
                                }
                        ]
val                     0 wei
```

Fig. 9. Product on sale by manufacturer



```
[vm] from: 0xAb8...35cb2 to: SupplyChain.shippedbymanufacturer(uint256,address) 0xd91...39138 value: 0 wei data: 0x9e7...c02db logs: 1 hash: 0x894...17e77
status                  true Transaction mined and execution succeed
transaction hash        0x8947befe371b0170bb26e9b8f207e6b4922904efd356918d72c891e9fa817e77
from                    0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to                      SupplyChain.shippedbymanufacturer(uint256,address) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                     80000000 gas
transaction cost        151764 gas
execution cost          151764 gas
input                   0x9e7...c02db
decoded input           {
                                "uint256 uin": "50851559",
                                "address shipTo": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db"
                        }
decoded output          {}
logs                    [
                                {
                                        "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
                                        "topic": "0xef1cb90ab53defb050ecdf1bec6f745742f554e1b09526e39f960d571cd1f3bd",
                                        "event": "EShippedByManufacturer",
                                        "args": {
                                                "0": "50851559",
                                                "1": "1648897696",
                                                "2": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                                                "3": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
                                                "uin": "50851559",
                                                "timeStamp": "1648897696",
                                                "caller": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                                                "receiver": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db"
                                        }
                                }
                        ]
val                     0 wei
```

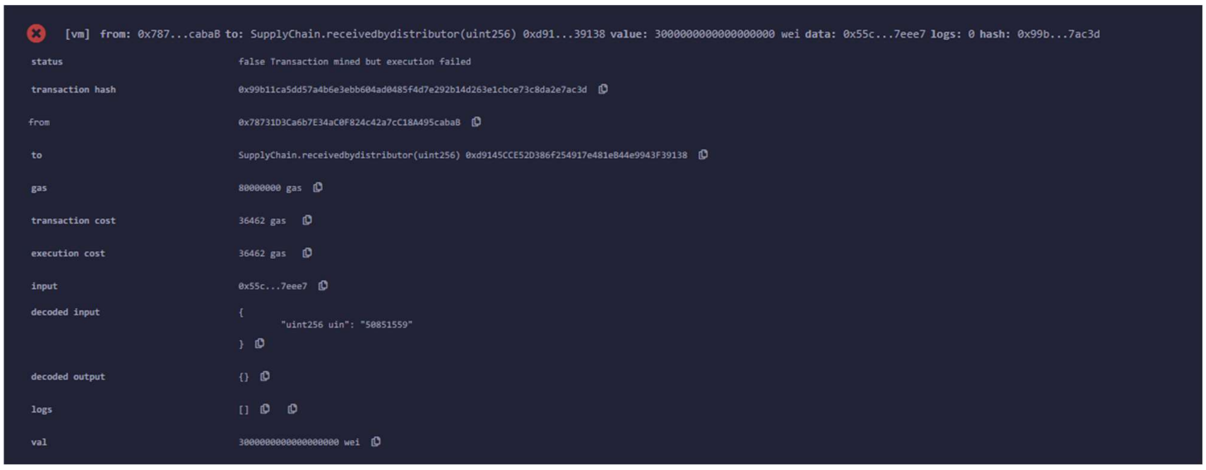
Fig. 10. Product is shipped to receiver's address

6.3. Ship Product to Next Entity (Distributor)

Off chain communication takes place between two parties for sale of product. Once the product is sold and shipped the event is emitted in blockchain. Sender will add the address of the receiver which will be encrypted in product hash. Figure 10 shows that the product state is updated and is currently in shipping state to receiver.

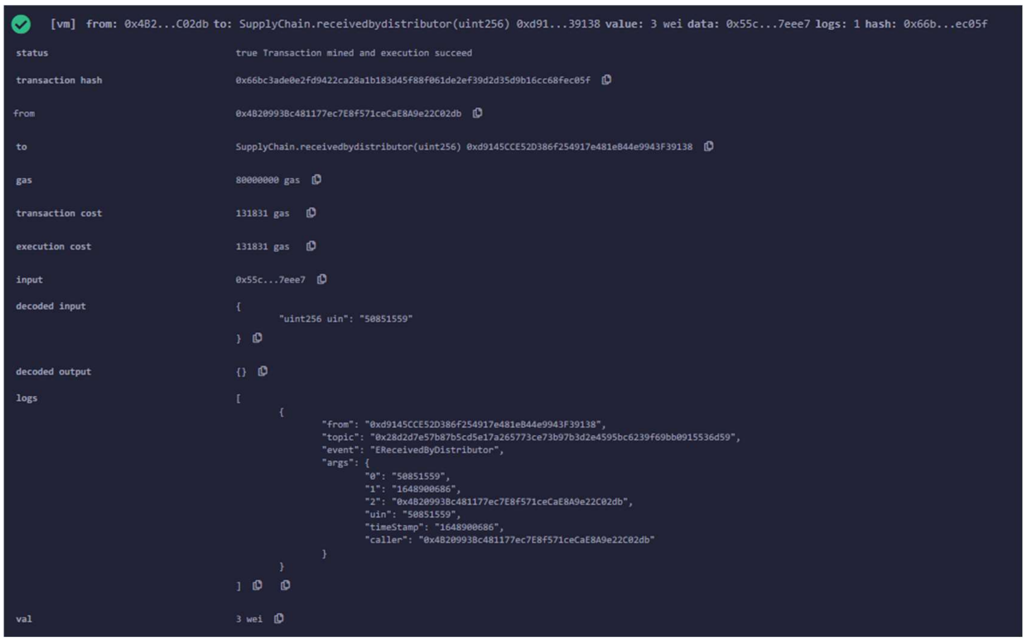
6.4. Delivery of Product to Wrong Address

During the process of process of shipping the product can get delivered to wrong address. In such scenario during verification if would be found and transaction won't be completed. Since the new hash generated during receiving purpose, will be compared to product hash added during shipping.



```
[vm] from: 0x787...cabaB to: SupplyChain.receivedbydistributor(uint256) 0xd91...39138 value: 3000000000000000000 wei data: 0x55c...7eee7 logs: 0 hash: 0x99b...7ac3d
status                  false Transaction mined but execution failed
transaction hash        0x99b11ca5dd57a4b6e3ebb604ad0485f4d7e292b14d263e1cbce73c8da2e7ac3d
from                    0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB
to                      SupplyChain.receivedbydistributor(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                     80000000 gas
transaction cost        36462 gas
execution cost          36462 gas
input                   0x55c...7eee7
decoded input           {
                                "uint256 uin": "50851559"
                        }
decoded output          {}
logs                    []
val                     3000000000000000000 wei
```

Fig. 11. Shows the failed transaction of product and ownership is retained by previous owner.



```
[vm] from: 0x4B2...C02db to: SupplyChain.receivedbydistributor(uint256) 0xd91...39138 value: 3 wei data: 0x55c...7eee7 logs: 1 hash: 0x66b...ec05f
status                    true Transaction mined and execution succeed
transaction hash          0x66bc3ade0e2fd9422ca28a1b183d45f88f061de2ef39d2d35d9b16cc68fec05f
from                      0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
to                        SupplyChain.receivedbydistributor(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                       80000000 gas
transaction cost          131831 gas
execution cost            131831 gas
input                     0x55c...7eee7
decoded input             {
                                  "uint256 uin": "50851559"
                          }
decoded output            {}
logs                      [
                                  {
                                          "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
                                          "topic": "0x28d2d7e57b87b5cd5e17a265773ce73b97b3d2e4595bc6239f69bb0915536d59",
                                          "event": "EReceivedByDistributor",
                                          "args": {
                                                  "0": "50851559",
                                                  "1": "1648900686",
                                                  "2": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
                                                  "uin": "50851559",
                                                  "timeStamp": "1648900686",
                                                  "caller": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db"
                                          }
                                  }
                          ]
val                       3 wei
```

Fig. 12. Successful verification of Hash

6.5. Delivery of Product to Correct Address

Once the product is received by correct recipient of product the transfer of funds from receivers account are automatically transferred to the previous owner resulting in transfer of funds and ownership of product refer Figure 12.

7 Conclusion

The developed method works for both multiple and single transporters, and includes a punishment and incentive mechanism to encourage all players to act honestly. The system also does away with the requirement for a trusted third party, instead using smart contracts as an escrow to automatically settle payments and distribute each participant's agreed-upon share once the item is delivered to the buyer. To meet the demand of delivering across multiple transporters, our decentralized solution uses a chain of contracts with no cyclic dependencies. For future work, depending on the product of application changes can make it compatible for multiple products. E.g., Vaccine, Agriculture products, Medicines

8 References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[2] H. R. Hasan and K. Salah, "Blockchain-based solution for proof of delivery of physical assets," in International Conference on Blockchain. Springer, 2018, pp. 139–152.
[3] Two party contracts. [Online]. Available: https://dappsforbeginners. wordpress.com/tutorials/two-party-contracts/
[4] K. Toyoda, P. T. Mathiopoulos, I. Sasase, and T. Ohtsuki, "A novel blockchain-based product ownership management system (poms) for anti-counterfeits in the post supply chain," IEEE Access, vol. 5, pp. 17 465–17 477, 2017.
[5] K. Biswas and V. Muthukkumarasamy, "Securing smart cities using blockchain technology," in 2016 IEEE 18th International Conference on High Performance Computing and Communications.
[6] A Review of Blockchain Technology and Its Applications in the Business Environment by Thomas Kitsantas Athanasios Vazakidis Evangelos Chytis at International Conference on Enterprise, Systems, Accounting, Logistics & Management July 2019
[7] https://ethereum.github.io/yellowpaper/paper.pdf
[8] http://myinstantid.com/szabo.pdf.
[9] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," IEEE Access, vol. 4, pp. 2292–2303, 2016.
[10] https://ethereum.org/en/smart-contracts/
[11] A. M. Antonopoulos and G. Wood, Mastering ethereum: building smart contracts and dapps. O'Reilly Media, 2018.
[12] The Keccak reference by Guido Bertoni, Joan Daemen, Michaël Peeters Gilles Van Assche, https://keccak.team/index.html.