UNIVERSITY of STIRLING

InDaChain: Transparency across the Supply Chain

Proof-of-Concept based on smart contracts

Pedro Herrera Lopez Guerrero

September 2019

Dissertation submitted in partial fulfilment for the degree of

Master of Science in Fintech

Computing Science and Mathematics

University of Stirling

Abstract

InDaChain is a proof-of-concept that benefits from blockchain capabilities to design a more transparent supply chain. The initiative is empowered by three main ideals: becoming a trustworthy source of information; create a decentralized system; and achieve a sustainable way of production. This project aims to connect two opposite ends across the supply chain: producers with the need to display a transparent way of production and customers interested in knowing the exact content of the goods they are acquiring.

The purpose of this document is to define a framework for future development based on the experience acquired by studying a real production process. Moreover, this text also provides a thorough overview of the basic concepts required to understand the logic behind blockchain and smart contracts, along all the technologies and processes required to interact with them. The results pave the way for constructing smart contracts that can help revolutionize the industry.

Attestation

I understand the nature of plagiarism, and I am aware of the University's policy on this.

I certify that this dissertation reports original work by me during my University project except for the following:

- The production process in detail, as well as the stakeholders involved, were mostly obtained from meetings with an involved party.
- The IDC screenshot showing the result of the parsed ABI is result from Tommaso Caramelli's work.

Signature **Date**

Acknowledgements

To my family, for supporting me my entire life. Words are not enough to demonstrate what they represent to me;

To my friends, in particular Pablo Soria who awaken the inner competitor and always kept me interested in computing science;

To my girlfriend for being my support at most difficult times and for teaching me how to reference properly;

To my former EY leaders Julio I. Hernandez, Felix Tarano and Jorge Hernandez who believe in me and gave me the opportunity to develop myself in a professional and competitive environment never letting me stay in a comfort zone.

To my program colleagues, specially Tommaso Caramelli and Manaf Safarini whom I develop the IDC project with. For making the dissertation time a one to remember and

To the University of Stirling and my Supervisor Andrea Bracciali for providing me with the right insights during the dissertation and throughout the whole MSc degree.

"I can accept failure, everyone fails at something. But I can't accept not trying"

Michael Jordan

Table of Contents

List of Figures

List of Tables

1. Introduction

InDaChain (IDC) is a Proof-of-Concept conceived along two colleagues of the MSc Fintech, consisting in a distributed application prototype comprised by a web/mobile front-end connected to a blockchain backend. The current project targets two main drivers: a transparent supply chain, and the support for those mindsets that embrace a certain type of alternative while producing or consuming a specific type of product, either organic, gluten-free or ethically produced.

With this decentralized application (dapp), each producer would be able to register its own data on the blockchain. Moreover, with the use of Smart Contracts along other technologies such as smart censors, products and processes can be certified while shipments can be tracked, providing trust across a whole supply chain. In the end, any potential end user can verify the origin of a product or the type of conditions required to produce a certain good.

The present document will first explain the general concepts of blockchain and smart contracts, as well as their risks and limitations. Then, a brief recap of the technologies involved in the development of the IDC smart contract, and requirements needed. Afterwards, the contract implementation and the main functionalities are also displayed. Finally, a conclusion of the whole work, providing an assessment and a framework that references potential future applications.

1.1 Background and general context

As Alvin Toffler once said, “the future comes fast”. As is happening across several industries, Supply Chain is an important sector that is becoming increasingly involved with digitalization and emerging technologies. From smart sensors that allow to track a freight in real time, to processing billions of custom documents required for commercial activities, the Supply Chain Business Model is evolving, paving the way to smarter and digitalized companies. It is estimated that by 2020, 50% of the industry business software will feature cognitive processing functionalities. [1]

At the same time, individuals have become more aware and cognizant about the impact that industrial systems present to our ways of living. As defined by the World Commission on Environment and Development, sustainability is the “development of the needs of the present without compromising the ability of future generations to meet their needs” [2].

Taking advantage of blockchain intrinsic capabilities such as immutability, traceability and accessibility, IDC offers the opportunity to have a transparent picture of how a certain good is produced, complying with specific requirements along the journey, from soil to shelf. This transparency is acquired by gathering certain data, which combined with smart contracts, can generate a secure, accessible and trustworthy source of information for a certain good.

For producers and retailers, it is a unique opportunity to record the way in which a product is made and eventually sold. Ideally, this situation will encourage more producers to avoid harmful and compromising practices. Also, help local producers that are left out of any potential competition against industrial organizations by showing a natural and authentic customized process. In the other end, customers will have a more complete picture of the goods they acquire, including the source of the ingredients used to produce a good, the journey it experienced and the impact to the environment and society.

1.2 Scope and objective

This document is focused on defining and detailing the blockchain framework of the IDC Proof-of-Concept. Other aspects as the general understanding of blockchain, smart contracts and the technologies implementations and requirements behind these tools are covered in detail.

It is important to note that a real study case was used for the implementation process. Information regarding a gluten-free oat process was studied and interpreted. To understand the general process in which oats are produced, two meetings were held with the process owner to fully learn the know-how of the production process and the main stakeholders involved.

At this early stage of the project, IDC is solely focused on developing the rationale behind the smart contracts that support the interaction with the blockchain whilst complying with the aforementioned framework. In this way, the current project intends to act as a “virtual simulator” where interested parties can see how real or simulated data can be deployed on an Ethereum test network and thus, managed using the tools incorporated within the dapp architecture paving the way for a potential expansion.

1.3 Risks & Challenges

- Establish a reliable connect the product interface with the blockchain to keep register of live data.
- Find the appropriate distribution patterns
- Lack of market opportunities that would lead to simulated data
- Dependence with the web application
- Truffle and Ethereum interfaces and environments in general are undergoing rapid development. Some things or servers may not work well during simulations. Specially (Truffle and Remix)

2. State of the Art

As previously stated, IDC intends to be a Decentralized Application that exploits the integral capabilities of a blockchain to offer the user a complete scene regarding the origin of a given product, from how it was grown or produced to the freight carrying it to the retailer shelf.

In this chapter, I will briefly introduce the evolution of Blockchain technologies to understand the functioning of this technology which will be the foundation of smart contracts, the machinery behind a dapp. I will highlight some advantages of using a Turing-complete programming language such as Solidity as well as some limitations itself. Also, I will cover on the basics to understand smart contracts and how they work. Finally, a high-level overview on how a Supply Chain works and a blockchain use case applied to this industry to fully understand how a functional business model would look like.

2.1 Blockchain

It happens that at a certain period of time, an invention or instrument completely revolutionizes the way we live; as it was with the automobile in the 20s and the Internet in the 90s, blockchain has the chance to be the latest conception which is already entirely disrupting our way of living.

To understand properly how the blockchain works and how it gains its main capabilities, it is better to start by knowing the first generation of blockchain and how it evolves into our current situation; the technological changes involved and applications and how it is applied into other industries such as the supply chain.

2.1.1 First Generation – Blockchain 1.0

In 2008, Satoshi Nakamoto issued a paper explaining the creation of an electronic peer to peer value transfer system which eliminates the need of the central figure in financial markets, commonly played by the banks [3]. This concept is currently known as Bitcoin.

To get the Bitcoin running, a not so widely known technology at that time was used: the Blockchain. A Blockchain is a peer-to-peer (P2P) distributed transactional database, commonly recognized as a Distributed Ledger Technology (DLT), that provides secure digital signatures by hashing transactions into blocks using timestamps. The distributed ledger is comprised by a network of machines with processing capabilities called nodes [3]. Every node will record an exact copy of the transactions comprised in order to verify whether the transactions are indeed valid, i.e. the senders have sufficient credit to be spent and thus, avoid a potential double-spending. This mechanism is called a proof-of-work (PoW) [4].

The PoW incorporates the need of solving an increasingly complex algorithm based on asymmetricity; i.e. it is extremely complex for one side to solve the original problem but surprisingly easily to validate the solution. In this way, the PoW accomplishes two main purposes: the prevention of double-spending transactions and the creation of new digital tokens that are rewarded to those servers in charge of processing the problem. This puzzle will be solved by brute force, hence computational power is key to solve it. The bigger the block, the harder an algorithm is to solve but the reward is also greater.

When a node finds a solution, it is communicated to the network where checkers will issue a consensuated agreement [5]. Then, the PoW consensus is complete on a block and the first node to solve the mathematical puzzle is rewarded a block for verifying the correctness of a transaction. This process is called mining. The block will be added to the previous computed block completing the mining cycle where each node can become a miner. Proof of Work enables a **distributed** and **consensuated** manner to verify each transaction at any desired time [6].

In blockchain, a block normally contains the generated unique hash value obtained from solving the PoW puzzle, this unique value is called Nonce. A block will also be comprised of the previous block's hash as well as the transactional (Tx) data (Figure 1). When a block is widely published into the blockchain, it will contain a unique set of data forming a continuing chain of **immutable** and **secure** passwords [3]. If any person intends to change any data in a block that is already mined, the nonce of this block will change, altering the established sequence with the subsequent blocks. In this way, if a block were to change, the hash of the following blocks should also be modified in a never-ending sequence the hash of the previous block, as well as the data corresponding to the transactions registered in the block.

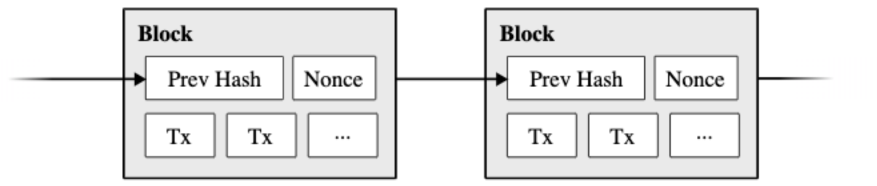


Figure 1: Block architecture (Source[3])

Blocks are protected by a strong security technique called public-key cryptography. This method enables the hash values to act as passwords for any transaction happening in the blockchain. In this system, each individual possesses a pair of cryptographic passwords, a Public Key and a Private Key [7].

The Public Key is a randomly generated address which is visible to every user connected to the blockchain. In the other hand, a Private Key is a secreted address, mathematically related to all addresses connected to a blockchain but that must be maintained in secret to validate transactions [3]. With this infrastructure, each transaction is protected through a digital signature which is sent to the "public key" of the receiver, and is digitally signed using the "private key" of the sender. In order to spend money, the owner of the cryptocurrency needs to prove his ownership of the "private key" (Figure 2).

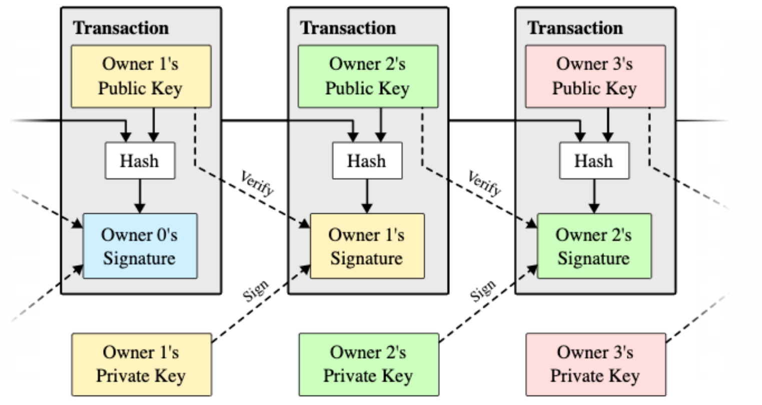


Figure 2:Transactions flow (Source [3])

While blockchain structure per se allows a distributed, consensus and immutable transactional ledger, the security offered by cryptographic means (hash) along the consensus mechanism thrusted into the network removes the need of a third party to verify the ownership of the transactions once double spending was overruled. This situation altogether brings one of the most vital properties of a blockchain: decentralization.

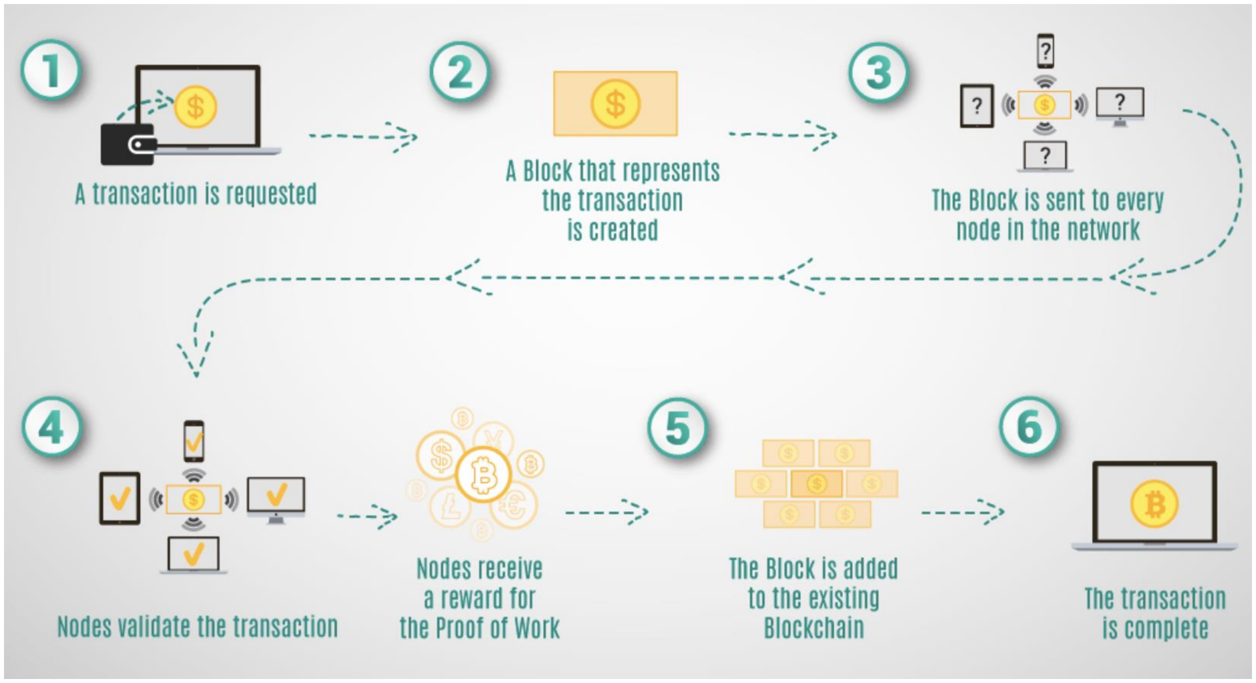

1
A transaction is requested
2
A Block that represents the transaction is created
3
The Block is sent to every node in the network
4
Nodes validate the transaction
Nodes receive a reward for the Proof of Work
5
The Block is added to the existing Blockchain
6
The transaction is complete

Figure 3: How the blockchain works? (Source[4])

2.1.2 Main limitations

Even though blockchain's disruption factor was undeniable, the first generation presented a group of challenges that, up to the date, are still existent. Issues related to sustainability, scalability, latency and security were and probably are still stated as genuine concerns. Yet it is precisely the blockchain consensus mechanism which is associated to these problems. The Proof of Work concept existed even before bitcoin, but it was this principle that helped the crypto asset revolutionize the way we can transfer value nowadays. In fact, probably the most important blockchains as we know them (Bitcoin and Ethereum) apply this type of consensus for the block generation and validation. However, this mechanism presents a huge deficiency that constrains the efficiency of this technology [8].

Sustainability: One of the main tangible issues surrounding Blockchain technologies is the computational power required to mine a transaction. The pessimistic side indicates that it could only get worse due to an increase in the level of transactions and a subsequent increase in the complexity of mathematical algorithms. The PoW mechanism basically facilitates a competition between miners that will reward the node with the bigger hardware capabilities. For example, to put things into perspective let's use Bitcoin operational numbers. Each Bitcoin transaction consumes 251 KWh of electricity which is enough to power 8.5 US homes for an entire day [9] or a British home for a month [10]. The entire Bitcoin network annual consumption surpasses the electricity consumed by 159 registered countries including Denmark and Colombia [11].

Scalability: The PoW consensus mechanisms, along other parameters such as the block-size, determine how fast a network can validate and confirm transactions. While the potential throughput transaction average in Bitcoin network and Ethereum is 7tps (transactions per second) and 15tps respectively [12], commercial payment networks can process thousands of transactions per second. Just to draw a comparison, if there is one, the tps average for other transaction processing networks are 2,000tps for VISA payments and 5,000tps for the Twitter social network [13]. Though a pragmatic solution would be increasing the rate of processing transactions, new imperfections could arise in the network. Security protocols could take a hit by those average-sized miners that would not be able to properly mine the upcoming transactions leading to centralization risks. As stated by Madisetti and Bahga [14] there is a theoretical trade-off between a blockchain processing speed and the level of decentralization that can be maintained within the network.

Latency: One of the aspects a blockchain protects is from double-spending, which is the result of successful spending of money more than once from one address. This makes latency a big issue in Blockchain currently. In a perfect world, making a block and confirming the transaction should happen in seconds, whilst also maintaining security.

Security: Blockchain and its based applications, which will be revised shortly, offer several advantages against web applications for example. Decentralization ensures that there is no a single breaking point where technical flaws or security breaches compromise either functionalities and/or sensitive information. Nevertheless, blockchain is vulnerable to certain type of security breaches. For example, by exploiting the PoW mechanism, an external entity can be able to garner enough share of the network to achieve 51% majority, hence running its own validations on upcoming transactions [15]. However, on the bright side, as processing requirements increase, the cost-benefit of performing an attack may not be beneficial to the criminal. Other flaws can be exploited if the programs that interact with a blockchain are poorly constructed.

2.1.3 Second Generation – Blockchain 2.0

Blockchain's next generation commenced as an attempt to overcome the limitations that the technology intrinsically entails. Several developers started to conceive different perspectives of the network which gave way to two critical innovations: a new consensus mechanism called Proof of Stake and smart contracts which will be explained further in detail.

With Proof-of-Work, the probability of mining a block depends on the computational capabilities of each miner [16]. Unlike the PoW where miners compete one another for solving mathematical problems, the Proof-of-Stake (PoS) presents a new paradigm. A new block is created in a deterministic way based on the wealth of the maker, which in this case are not called miners but forgers as previously mentioned. This means that in the PoS system there is no block reward per se, so the miners end up taking the transaction fees. For example, someone holding 15% of the Bitcoin can mine 15% of the "Proof-of-Stake blocks" [16]. This theory depends upon the theory of the larger "stake" anyone holds within the network, the lower chances to get breached due to the already high stakes if performance is optimal.

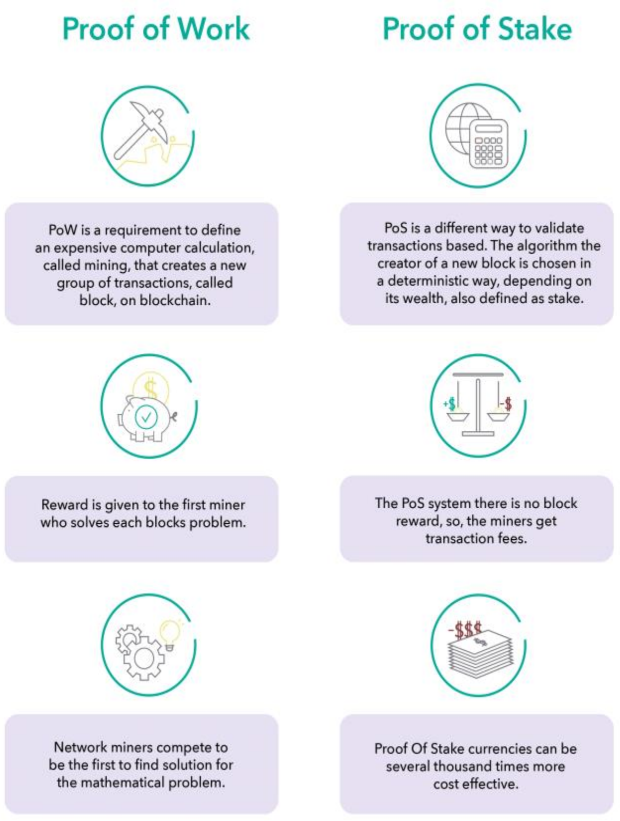

Proof of Work
Proof of Stake
PoW is a requirement to define an expensive computer calculation, called mining, that creates a new group of transactions, called block, on blockchain.
PoS is a different way to validate transactions based. The algorithm the creator of a new block is chosen in a deterministic way, depending on its wealth, also defined as stake.
Reward is given to the first miner who solves each blocks problem.
The PoS system there is no block reward, so, the miners get transaction fees.
Network miners compete to be the first to find solution for the mathematical problem.
Proof Of Stake currencies can be several thousand times more cost effective.

Figure 4: PoW vs PoS comparison (Source[16])

From an economic standpoint, a PoS mechanism can constitute an important countermeasure to the PoW protocol. The stake possessed by each forger can yield directly proportional results in terms of security and efficiency at the moment of creating new blocks. As Saleh states [17], the more modest the block reward schedule becomes, the laxer the necessary restriction becomes. The benefit of a modest block reward arises in part because block rewards enable validators to accrue vested interest on lagging branches which in turn creates an incentive for those validators to persist disagreement.

Unlike the PoS which is still struggling to function properly at a large scale, the second generation of blockchain gave birth to one of the main disruptors in today's world: smart contracts.

To understand smart contracts, it is important to understand first the technology on which these programs are run: Ethereum. In 2013, Vitalik Buterin issued a paper that described a decentralized network capable of processing payment transactions within its own internet browser, using a built-in coding language. In words of Buterin, Ethereum's mission is to create a network of private computers that run various applications without a third party [18] where parties agree to collaborate into common goals or outcomes including exchange of goods and service.

Ethereum is a public, open-source, Blockchain-based distributed software platform that allows developers to build and deploy decentralized applications. Although Bitcoin is recognized as the first blockchain technology to go big, Ethereum presents a vast array of characteristics that make this platform a more enticing option for the project to be developed, especially with the inclusion of smart contracts [19].

Supported with a built-in Turing-complete programming language, Ethereum allows the users to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions. Additionally, the network consensus mechanism deployed by the nodes is driven around this type of functions [18].

Moreover, the categorical structure of an Ethereum protocol is slightly different than the one used for mining Bitcoins. Each Ethereum node serves as an Ethereum Virtual Machine (EVM), a provisional environment that provides the users with capabilities of executing a smart contract [20]in an open and secure manner. In this way, an EVM removes the strict need for having huge hardware capabilities to store transactions but only enabling executing features.

Although Ethereum is not ready to operate live under a PoS mechanism yet, Smart Contracts allow the Russian blockchain to overpower Bitcoin in terms of technology and potential. Several authors including Swan [21], have identified some potential disadvantages when comparing Bitcoin to Ethereum. No matter which version is being used, Ethereum offer enticing advantages, when compared to the Bitcoin Blockchain.

In terms of sustainability, scalability and latency, Ethereum provides better indicators than its more famous peer. For example, when comparting the amount of electricity needed to mine a transaction in an Ethereum network is almost 1/10 of the energy required to mine a Bitcoin one. The following table (Table 1) presents the most important metrics of the network in terms of sustainability:

Description	Value
Ethereum's current estimated annual electricity consumption (TWh)	7.1
Annualized global mining revenues	$1,821,485,047
Annualized estimated global mining costs	$709,855,697
Current cost percentage	38.97%
Country closest to Ethereum in terms of electricity consumption	Bolivia
Estimated electricity used over the previous day (KWh)	19,448,101
Implied Watts per MH/s	4.707
Total Network Hashrate in GH/s (1,000 MH/s)	172,150.00
Electricity consumed per transaction (KWh)	24
Number of U.S. households that could be powered by Ethereum	657,274
Number of U.S. households powered for 1 day by the electricity consumed for a single transaction	0.82

Description	Value
Ethereum's electricity consumption as a percentage of the world's electricity consumption	0.03%

Table 1: Ethereum key indicators (Source[9])

In terms of latency and scalability, when deploying a smart contract in an Ethereum public network, there is approximately 13 seconds from the moment the transaction is executed to get a proper validation in the entire network. For example, as of the end of August 2019, an Ethereum network computes a transaction in slightly more than 13 seconds (Figure 5).

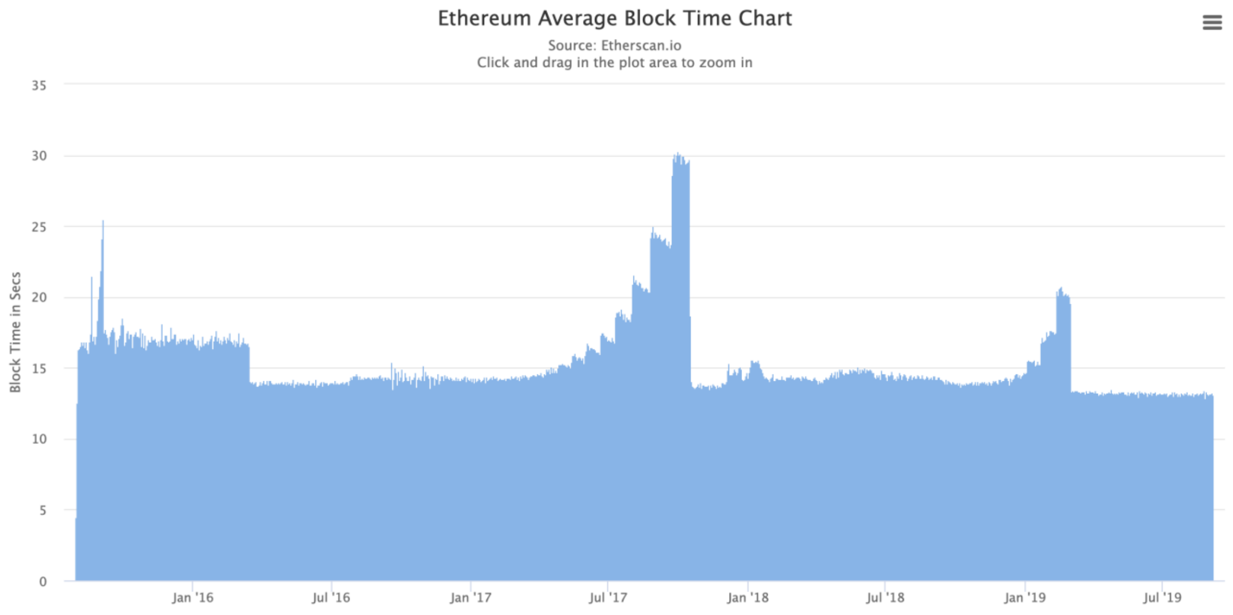


Figure 5: Ethereum blocktime chart (Source[22])

As previously mentioned, Ethereum does not operates yet under a PoS consensus mechanism. However, at the start of 2019, Vitalik Buterin, Ethereum creator, and Virgil Griffith [23] issued a paper to introduce Casper, an Ethereum Proof of Stake mechanism that combines PoS with Byzantine fault tolerant algorithm that rely on consensus theory. By shifting the consensus mechanism, Ethereum will substantially reduce the energy required for creating and validating transactions. It is believed that with these implementations, the network will be able to process in the order of hundred transactions per second, improving both sustainability and scalability [24]. Although far from being finished, the Casper PoS aims for a more efficient process limiting the energy consumption and thus working in a greener manner. Nevertheless, it is outstanding the way in which Ethereum took advantage of economic theory and defined the framework on which a gas prioritizes the forging process. Without knowing, Buterin had already envisioned a system that prioritized significant transactions by optimizing computational costs.

2.1.4 Smart Contracts

Perhaps the most important generational leap from the first generation to the second one is the inclusion of smart contracts. A Smart Contract is the basic group of programmable functions behind the applications and programs deployed in Ethereum [18]. These cryptographic “boxes” contain lines of code that automatically execute any agreement based on the fulfilment of certain conditions required by the contract itself. Let’s try a simple example: Transporter A carriages dairy products into Storage X. These products must maintain a certain temperature during the transportation and need to arrive at an agreed time. If these conditions are met, Storage X will pay a defined amount of Ether to Transporter A. If the conditions are met, the contract will automatically release the payment, but if any condition is not fulfilled, the payment will not be performed (Figure 6).

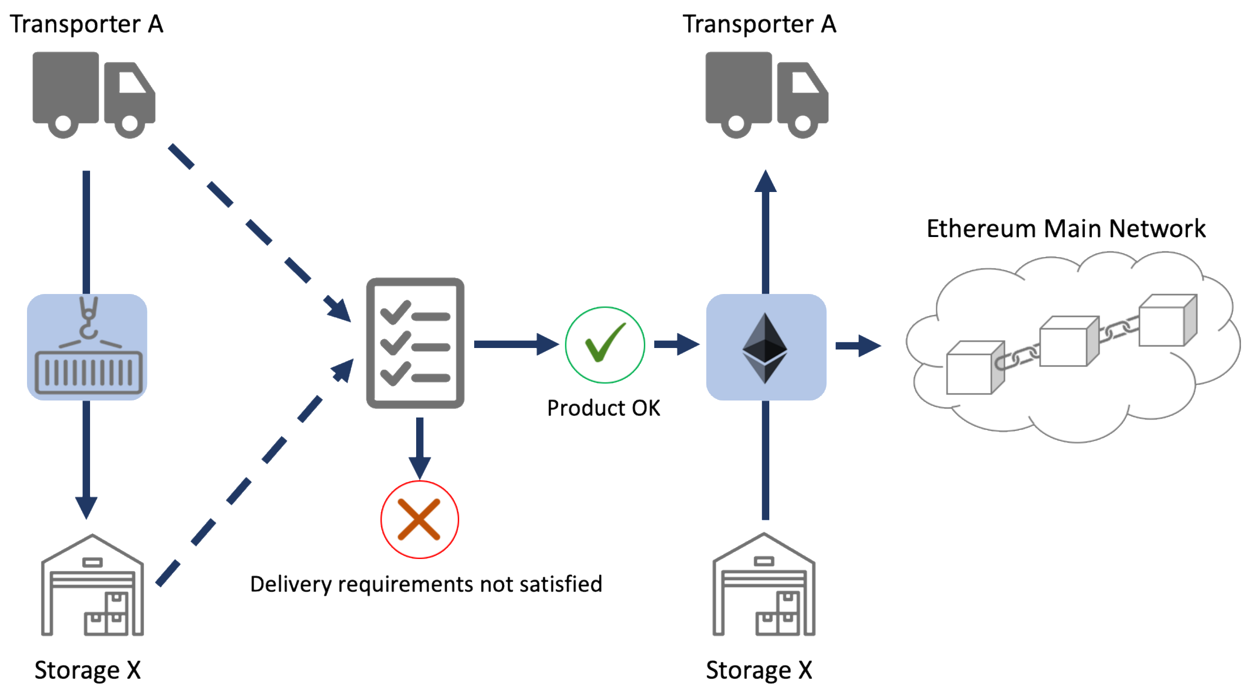


Figure 6: Smart Contract logic

With smart contracts, the value offered by blockchain changed from being a decentralized network that allows the transfer of a digital currency to bringing the user with the opportunity of programming the transactions itself. As stated by Tsao [25], by linking smart contracts to a blockchain, a potent processing network is created which also be decentralized, censorship-resistant and immutable.

The "revolutionary aspect" of Ethereum is the ability that allows users to code their own functions and deploy them under specific circumstances as explained in Figure 6. The programs ran on an Ethereum network consist of functions executing transactions. Hence, a program executed on an Ethereum network ought to pay a miner for every transaction that is successfully verified. In addition, the cryptocurrency associated to the network, Ether, is not only used as a digital currency to be transferred between individuals, but is literally the gas that empowers the blockchain. These decentralized smart programs combined can virtually allow any program that exists today to be adapted for the Ethereum network [25].

A smart contract consists of three main parts: a program code, a storage file, and an account balance. Any user can create a contract by posting a *transaction* to the blockchain that will cost a certain amount of gas.

The program code of a contract is fixed when the contract is created, and cannot be changed, while its storage file, is stored on the public blockchain. A contract's program logic is executed by the network of miners who reach consensus on the outcome of the execution and update the blockchain accordingly. While the contract's functions can be invoked by any user or from another contract, the contract may just retrieve data or update the storage file. Also, a contract can execute value transfers between different user accounts [26].

Gas is a concept that has been previously mentioned and is critical to understand smart-contracts operation and their relation with the Ethereum ecosystem. Gas is a resource used in Ethereum to manage the implementation of smart contracts [27]. It is mainly used by a contract as a "currency" based on Ethers to pay fees to the forgers (Ethereum miners) for their services. Every time a contract is deployed in a real Ethereum network it spends an amount of money to pay the forgers so that transaction can be published in the Blockchain.

For example, in a world without ETH, to run code in a decentralized server (e.g. Amazon Web Services - AWS) the user will pay for the infrastructure to run the code, in the same way, a user will pay the forgers to run the contracts. The name is even ironic as a real-life comparison to how gasoline affects a car performance: higher speed consumes more gasoline than driving conservatively.

In this way, every time a contract is executed, a certain amount of gas is spent, and also every time a transaction gets executed the entire PoW mechanism validates it. The value of a transaction can be measured in the smallest denomination of Ether called Wei (10^{18} Ether), GWei (10^{10} Ether), Finney (10^{3} Ether), or Ether itself [28].

There are two important concepts to consider when calculating the amount of gas spent. The first one is the gas Price, which is the amount of Wei a sender is willing to pay per gas unit to get a transaction processed; i.e. the price at which gas is being traded. If the gas Price is not properly valuated, miners will not be compensated accordingly, hence, the transaction will not be validated which will result in desertion. A higher gas Price is most costly to the sender in terms of real Ether, but is also more likely to be selected by forgers for inclusion [18].

Secondly, there is a gas Limit, which can be defined as the units of gas that a transaction can consume. By using appropriate thresholds, developers can ensure that their contracts can be deployed steadfastly on a network [29].

The amount of gas consumed by a contract is variable. There are innumerable different operations that can be executed by a contract, and each one of them carry a different cost depending on the task performed [18]. Also, a certain amount of gas consumed by each transaction is used to pay for the storage on the blockchain.

In the following example (Figure 7), a contract is trying to call a function called 'DoMath', which adds, subtracts, multiplies and divides a certain integer. The Price paid for each unit of gas is set by the developer at 300 Wei, while the gas Limit is set at 10. At the moment the node runs all the mathematical functions, it will halt the operation at 6 gas, since the subsequent command will not possess enough funds to pay for the operation. Should the contract run the multiplication function, gas Limit would exceed the established limit. No other code will be executed inside the function. In this case, 10 units of gas will be paid at no more than 300 Weis per Gas unit.

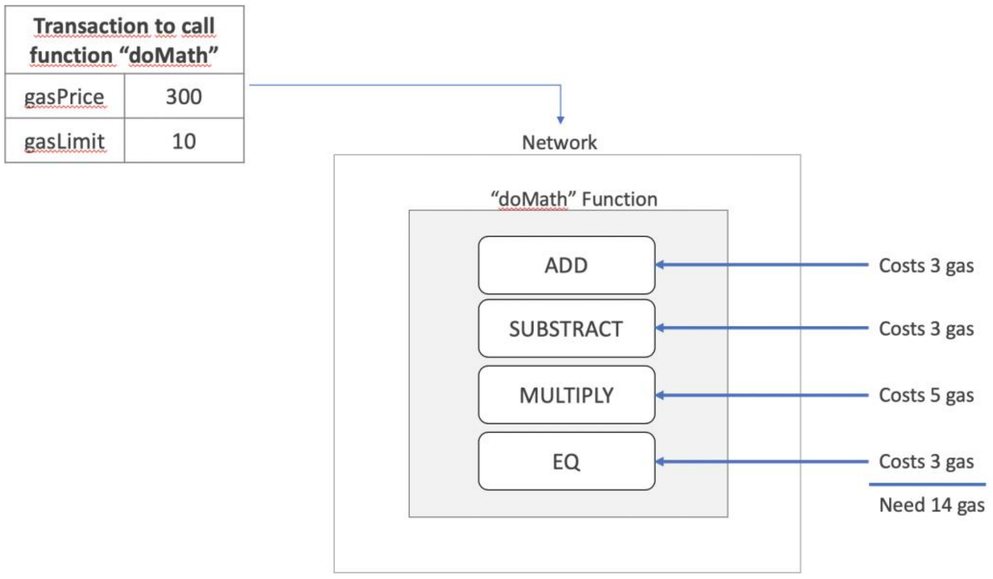


Figure 7: Insufficient gas funds

In the other hand, in the following figure (figure 8), the gas limit is set at 20; hence all transactions could be executed. However, the contract will not consume the whole 20 Gas units in Wei; at 14 Gas the transaction will be complete thus only 4,200 (14*300) Weis will be spent.

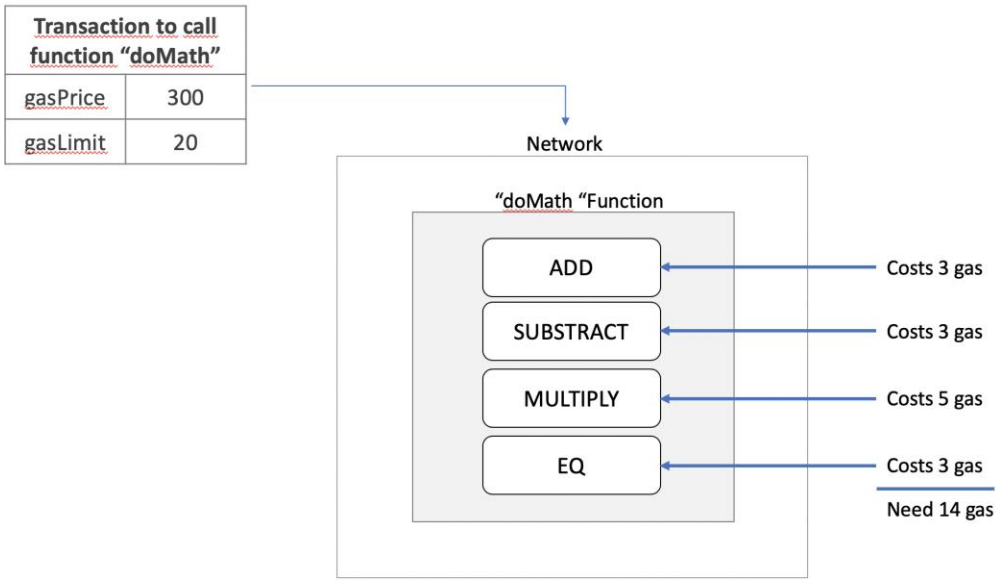


Figure 8: Sufficient funds

Hereby lies the importance of selecting the proper functionalities for a contract. For example, if a fetch function is executed it may only spend a fixed amount of gas. However, if a contract contains a loop function, the amount of gas spent can vary. A function is more expensive depending on the tasks deployed by each contract.

One situation worth considering is that apart from the main Ethereum network, there are different test networks where developers are able to test their contracts under development. Deploying a contract in the real Ethereum network incurs in real gas costs translating into real money used to deploy a contract.

The three main test networks are Ropsten, Rinkeby and Kovan. All of them present differences among them. Nevertheless, they share one critical trait: the ethers used within these networks are forged for the sole purpose of testing; i.e. they are not worth real money. So, these networks are mainly used to test a contract's performance by getting free ether and using virtual gas [30].

The versatility of Smart Contracts allows the interaction with several programming languages that are Turing-complete such as Solidity, Serpent and LLL. This level of adaptability allows Smart Contracts to perform highly complex functions whilst adding awareness and stability [31]. This flexibility allows Ethereum users to build on top network applications known as Decentralized Applications or dapps. The holistic implications of dapps are not limited to a single context but is a complete game changer from any perspective. Being able to build and deploy in Ethereum means that any service can be decentralized.

A dapp is a smart contract executing itself within a blockchain network. It fills a typical role played by a web application but adding some special characteristics [15]:

1. Blockchain's decentralization nature allow smart contracts and thus the DAPP to run **autonomously**.
2. All the information recouped as a result of a dapp activity is **traceable** and **verifiable**.
3. Cryptographic properties along the consensus mechanism ensure the DAPP with a secure layer to a certain extent. Some limitations will be covered shortly.
4. Probably the most important contribution is the **stability** provided by a blockchain. Each node is fully connected and communicated to every other point in the network. In addition,

each node will store smart contract making this structure almost failure free in terms of operation.

In the last years, a group of successful dapps have achieved to bring a functionality beyond a transaction. Startups like steemit, storj or openbazaar have established themselves as integral leaders of an emerging industry. In addition, since there is no centralized client server architecture, this type of systems has become a more secure option.

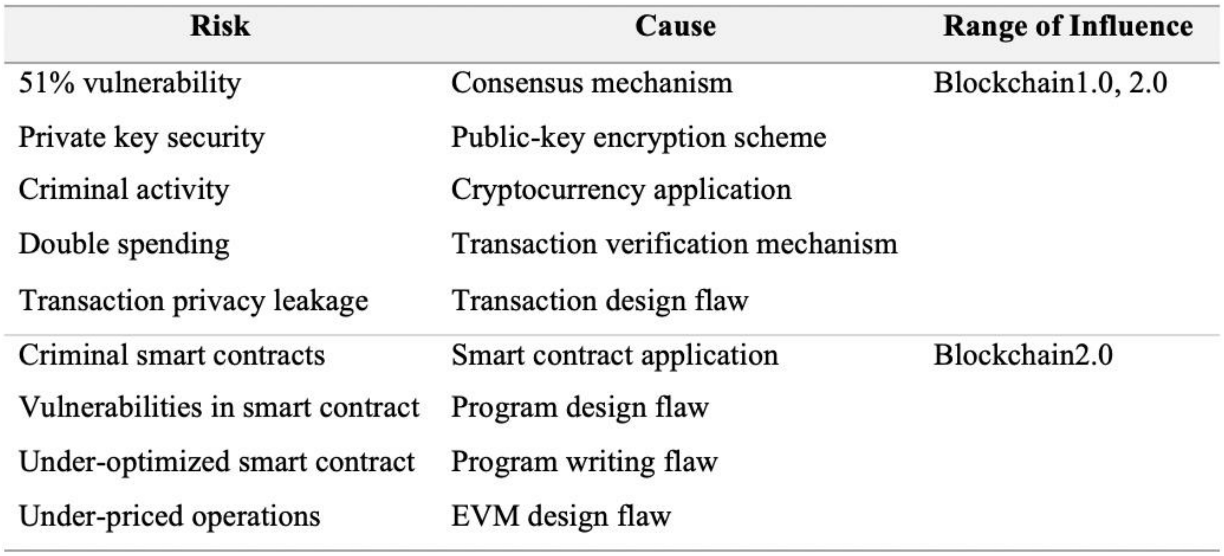
As previously revisited, a dapp containing complex functions such as loops and iterative approaches, can become a problem since it can be really expensive to run. It is uttermost important to highlight that the individual that runs the transaction, pays the gas price for it. For example, were Twitter a dapp, each time a user wants to send a tweet, it ought to pay a certain amount of gas. This situation can limit the scope of applications to be developed.

2.1.5 Risks in Smart Contracts

Vulnerabilities are persistent in all platforms and networks across the globe even bringing down the most sophisticated defences down. Blockchain is no different and present some risks that are inherent to its own nature. Therefore, it is important to acknowledge the potential risks that are present in a Blockchain network.

Not only functionalities and features are different from one blockchain generation to the other. While Ethereum enjoys a wide array of possibilities thanks to the Touring-code capabilities, it also triggers potential security complications due to its complexity. Since smart contracts are linked to an Ether balance, they are a common target for criminals.

Hacks targeting blockchain include the Decentralized Autonomous Organization (DAO) attack which in 2016 claimed roughly 3.6 million Ether [32] and Parity multi-sig attack [33] have led to millions of dollars in losses. Li [20] enlists nine blockchain risks that are present in the technology (Table 2). Five of them attain both generations and are mostly related to the operation mechanism of the network. However, there are four critical risks that affect exclusively the Blockchain 2.0 specifically vulnerating the development, deployment and execution of smart contracts.



Risk	Cause	Range of Influence
51% vulnerability	Consensus mechanism	Blockchain1.0, 2.0
Private key security	Public-key encryption scheme	
Criminal activity	Cryptocurrency application	
Double spending	Transaction verification mechanism	
Transaction privacy leakage	Transaction design flaw	
Criminal smart contracts	Smart contract application	Blockchain2.0
Vulnerabilities in smart contract	Program design flaw	
Under-optimized smart contract	Program writing flaw	
Under-priced operations	EVM design flaw	

Table 2: Smart Contracts Risks Taxonomy (Source [20])

○ **Criminal Smart Contracts (CSC)**

This flaw is presented in the deployment and execution of a smart contract. It is conceived when an attacker compromises the functionality of a smart contract to perpetuate a crime, most commonly related to leakage or theft of privileged information, theft of private keys and the so called “calling-card” crimes [34]. Li [20] provides a graphic example (figure 9) of a CSC in which a perpetuator (P) is contacted by a contractor (C) to steal a targeted account (A) information. P uses an SGX instruction code via HTTPS connection to confirm that the CSC code is able to retrieve the private key of the targeted account after generating a private / public key pair to interact with A. If the CSC verifies successfully the information using the generated keys, it will send the data to C who will validate the information and will send a compensation to P for the services.

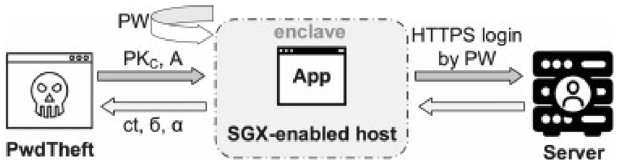


Figure 9: Execution procedure of Password Theft (Source[20])

- **Vulnerabilities in smart contract**

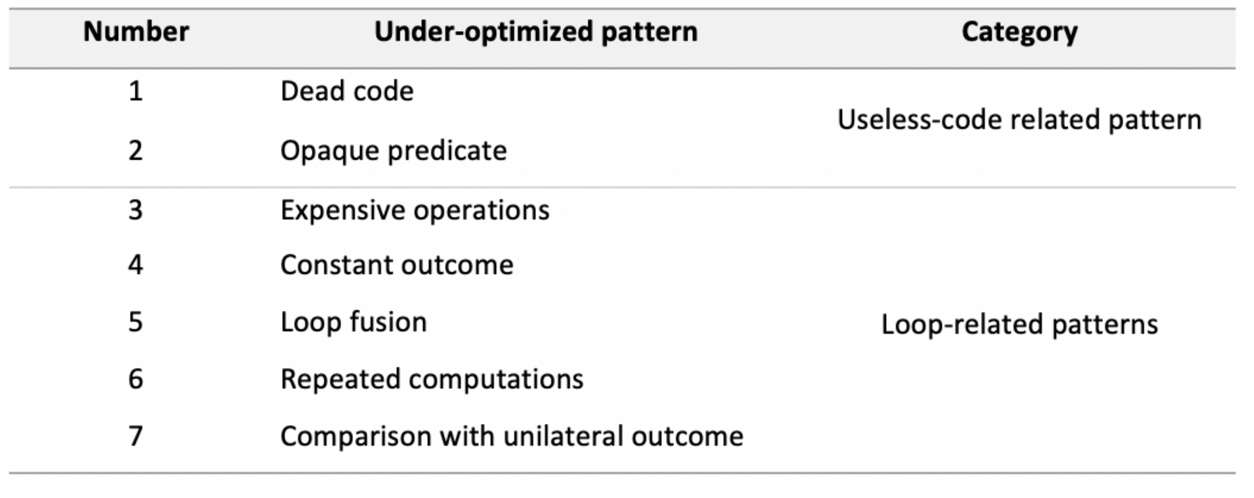
"With great powers come great responsibilities". As powerful tool a smart contract can be, the level of vulnerability is high as well. It depends on the strength, integrity and structure of the code to be executed in an effective manner. The risks of having a weak written SC is being exposed to tampering attacks in order to corrupt the desired instructions, in most cases, to deviate the final destination if the resources involved [35]. The following table (3) defined by Atzei, Bartoletti and Cimoli [35] describe the taxonomy of the main identified vulnerabilities in SC based on the level on which an attack is introduced: Programming language, EVM or network.

Level	Cause of vulnerability
Solidity	Call to the unknown
	Gasless send
	Exception disorders
	Type casts
	Re-entrancy
	Keeping secrets
EVM	Immutable bugs
	Ether lost in transfer
	Stack size limit
Blockchain	Unpredictable state
	Generating randomness
	Time constraints

Table 3: Smart Contracts Vulnerabilities Taxonomy (Source [35])

- **Under-optimized smart contract**

As previously explained, when a user interacts with a smart contract deployed in Ethereum, a certain amount of gas is charged. Unfortunately, the development and thus the deployment of some smart contracts are not adequately optimized. Chen et al. [20] identify 7 gas-costly patterns and group them into 2 categories (as shown in table 4): useless-code related patterns, and loop-related patterns.



Number	Under-optimized pattern	Category
1	Dead code	Useless-code related pattern
2	Opaque predicate	
3	Expensive operations	Loop-related patterns
4	Constant outcome	
5	Loop fusion	
6	Repeated computations	
7	Comparison with unilateral outcome	

Table 4: Under-optimized Solidity code patterns (Source[20])

The same group of authors, propose a tool named Gasper, which can automatically discover 3 gas-costly patterns in smart contracts:

- Dead code: those operations that are deployed in the blockchain but are never executed causing a gas consumption.
- Opaque predicate: the presence of these statements originates useless operations guzzling additional gas.
- Expensive operations in a loop

o **Under-priced operations**

Gas is a valuable resource that can be manipulated by an attacker to provoke unwanted behaviour in a victim's smart contract (e.g., wasting or blocking funds of said victim) [27]. When the Gas Price for a transaction is set in a low threshold, an attacker can call for the contract several times in one transaction causing a desynchronization in the network. This umbrage can be interpreted as an Ethereum Denial of Service (DoS) attack.

Some pioneer work has been done to improve the efficiency of blockchain. Zyskind, Nathan and Pentland [36] proposed a lightweight blockchain architecture to protect personal data. They improved the efficiency of blockchain by using off-chain data storage and heavy processing. Only references to data and lightweight processing tasks were handled in the blockchain. Paul, Sarkar and Mukherjee [37] proposed a new scheme that could lead to an energy-efficient Bitcoin. The authors modified added some extra bytes to the newest block header and utilize the timestamp more effectively.

Moreover, since developed dapps are built using an open source code like Solidity, it is highly important to follow best practices to avoid leaving any hacking opportunity in the contract code [28]. Conceptually, one can think of a contract as a special "trusted third party" – however, this party is trusted only for correctness and availability but not for privacy. In particular, a contract's entire state is visible to the public.

2.2 Transparency & Traceability in the Supply Chain

One of the most intriguing aspects of the Ethereum platform is the flexibility offered by the smart contracts' architecture. They can "communicate" between them, adapt to the requirements of the user and can be deployed and tested immediately without losing any of their properties. Not stopping there, due to the extensive amount of open code available, developers have begun to adopt more comprehensive dapps. Decentralized applications have opened a revolving door of options for several sectors to implement real business processes into the blockchain with the usage of smart contracts. Different sectors such as financial services, healthcare, education and voting E-Systems [38] have benefited from this technology.

For logistics and supply chain, it is no different story. dapps allow transactions and processes to be autonomous, traceable and secure [15]. These characteristics offer an enticing environment for these sectors. A decentralized P2P system in such a bureaucratic environment such as this one can help mitigating corruption and potential fraud whilst optimizing the services provided. The data comprised in the application is not owned by one organizational or govern body but its shared between several organizations. Also, the immutability inherent to smart contracts helps to constitute a **tamper-proof application**.

Blockchain set of competences (decentralization, autonomy and immutability) facilitate the creation of a **transparent and traceable supply chain**. Each individual operation or interaction, such as the provision of a new employee or the recording of outgoing stock, is perfectly recorded and archived. Information can be shared in a real time and accessible manner where users can visualize processes, places and materials used in the production cycle of a good or service. In this context, audit and certification process are also integrated. Auditing is thus as simple as joining the blockchain network, as this allows one to “replay” the operations of the past in order to build a correct model of the present. As in a domino effect, every individual interacting with a certain blockchain will have access to a product’s provenance.

Provenance is a quite an enthralling concept. It can be defined as the true origin of a product, englobing, what it contains, how it was produced and by who. To fully know the provenance of a product, transparency and traceability must exist at a high extent. Blockchain facilitates both circumstances by creating a distributed and immutable ledger.

Several occurrences around the globe have been originated due to opacity in the provenance of a product. It is a problem that can affect all type of goods and services, from buying a Picasso replica for millions of dollars [39] to the 2013 horse meat scandal in Ireland and the UK where 85% of the meat sold in big supermarkets showed presence of horse DNA [40].

Just like Provenance, which will be shortly explained, there are several other initiatives trying to bring a more transparent supply chain. Everledger [41] is one example. This company, in cooperation with Barclays, created a blockchain to certify that the diamonds that they are producing are not consider “Blood diamonds” ensuring that no human or ethical rights were infringed during their production.

2.2.1 Supply Chain / Smart Contract Use Case: Provenance

One of the most successful use cases in the supply chain industry is Provenance. Provenance is a startup originated in the UK that bases their core operations in the use innovative technologies like the blockchain to create a more transparent supply chain and garner trust around a certain product. As Jessi Baker, Provenance's CEO mentions [42], Provenance embraces sustainability by knowing where the products they offer come from. This benefits businesses and customers that embrace certain type of ideology, such as avoiding products that involve environmental or human harm.

Provenance works with different stakeholders to ensure that its main mission is being accomplished: enable every product to come with an open, secure record of its journey and creation. The main stakeholders involved in this process are [43]:

- Producers or manufacturers
- Registrars, or accreditation service organizations
- Standards organizations (e.g., Fairtrade)
- Certifiers and auditors
- End consumer

By being based on blockchain technologies, Provenance exploits the so much mentioned blockchain capabilities to build a transparent supply chain and build trust in the products they offer. The main premises in this case are traceability and security.

Being traceable means that all the interactions – inputs and outputs – across the whole process are perfectly auditable. In this way, any transaction related to a given product can be challenged. This level of traceability facilitates the creation of diverse programs for suppliers where inspections carried out by a certifier or auditor can be easily accessible. Through these programs, organizations can openly inform certifying agencies about animal testing, biodynamics or human labour conditions; production parameters to certify production capacity or specific product attributes or manufacturing conditions that need to be met [43].

Security in several contexts can be translated to trust. The security in Provenance is mostly enabled through the Public and Private Key authentication protocols. It enables a platform where the origin, quality and quantity of a product can be assigned and verified. An organization or a client can access the registration program and thus link their real-life data to the Provenance profile by using a private key. Should an authority commands to inspect certain traces of a given good, the public key will work in both ends to ensure a secure record [43].

Beyond the cryptographic infrastructure, establishing secure bridges between the real and digital world is also critical. By implementing QR codes and Near-Field Communication Tags (NFC) Provenance enables a user interface that keeps tracking in a secure way. [44]

Furthermore, the products we acquire from any supermarket or even online stores go through a fairly complex pathway. Several actors are involved throughout the cycle making the tracking of the whole production process a monumental task. That's when we can ask, at what extent can we be sure that the label in the products we buy is representative to its true provenance?

If we are able to know how the whole production process is undertaken, the situation might be very different. Opportunity areas in planning (demand forecasting, inventory management) and management (data sharing and product traceability, managing risks and disruptions, transparency and building trust and reputation) are enticing, especially when another disruption factor like Machine Learning is thrust into the equation.

Without decentralized consensus, the party providing centralized consensus often enjoys huge market power (e.g., a third party with data monopoly). And traditional resolutions by third parties, such as courts or arbitrators, involve high degrees of human intervention that are less algorithmic, potentially leading to greater uncertainty and cost. Smart contracts can increase contractibility and facilitate exchanging money, property, shares, service, or anything of value in an algorithmically automated and conflict-free way.

3. General Requirements

In this section, the main technological and program specific requirements used to develop the smart contracts will be discussed. Furthermore, it will detail the main functionalities and roles of each program used. It is important to consider that working with Ethereum, means that from the moment of the first interaction, a whole network of computers will be involved, either to validate an Ether transfer or simply to invoke or store any sort of transactional data required by the user.

In the last chapter, I briefly mentioned the existence of Ethereum test networks that developers use to assess those smart contracts that are still in development. For testing the IDC smart contract, the Rinkeby Test Network was used. This network uses a consensus algorithm called “Proof of Authority” [45]. With this type of mechanism, the user must authenticate to receive worthless ethers from a faucet [30].

3.1 Hardware

To develop the contracts, a MacBook Pro was used. 2.3GHz quad-core Intel Core i5 processor with 128 MB of eDRAM; 16GB of RAM, 512 GB SSD with Intel Iris Plus 655 Graphics Card.

In general terms, any PC with basic capacity should handle smart contracts deployment on an IDE such as Remix. A robust web connection is stringently required.

3.2 Technological requirements

Before describing the functional process of how to put a blockchain into full operation, there are some considerations to integrate the entire functionalities in a holistic manner.

3.2.1 Remix

Remix is an Integrated Development Environment (IDE) that acts as an online code editor that allows the user to write and test Solidity code directly on the browser. Remix can be accessed through any modern browser through the URL https://remix.ethereum.org/

Remix offer important advantages by supporting built-in smart contract development features such as testing, deploying and debugging by hosting a “fake” EVM in its own backend.

As of the writing of this document, Remix layout (Figure 6) consists of an icon panel where different modules and plugins can be enabled, a side panel that will represent the Graphic User Interface for enabled modules and functions, the main panel where the main code can be modified and the terminal where results of GUI's interactions will be shown [46].

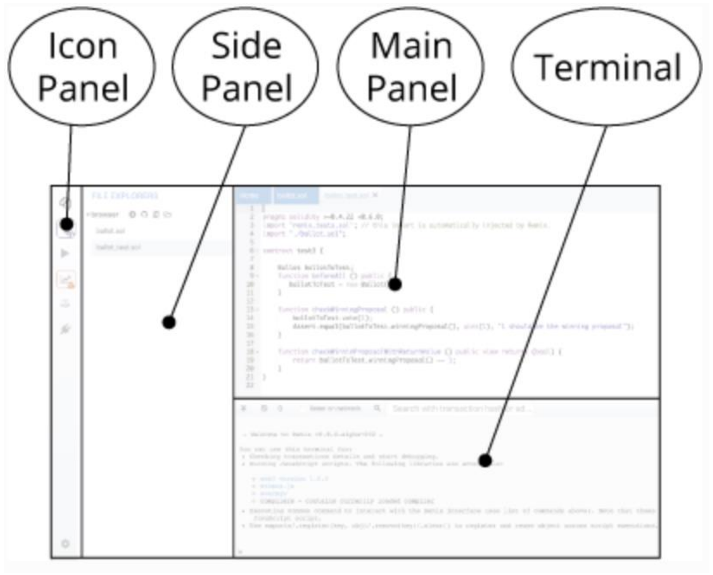


Figure 10: Remix Layout (Source[46])

The compiling task in Smart contract development is essential. When a contract is run, a compiler script will generate two additional files: the first one, is called bytecode, an array of characters that represent the actual Solidity code that is going to be stored and executed on the Ethereum network; the second file is called the Application Binary Interface (ABI) that will be deployed to the Ethereum network (figure 11). The ABI is essentially a translator between Solidity and JavaScript languages since the latter code has no ability whatsoever to interact with the bytecode that

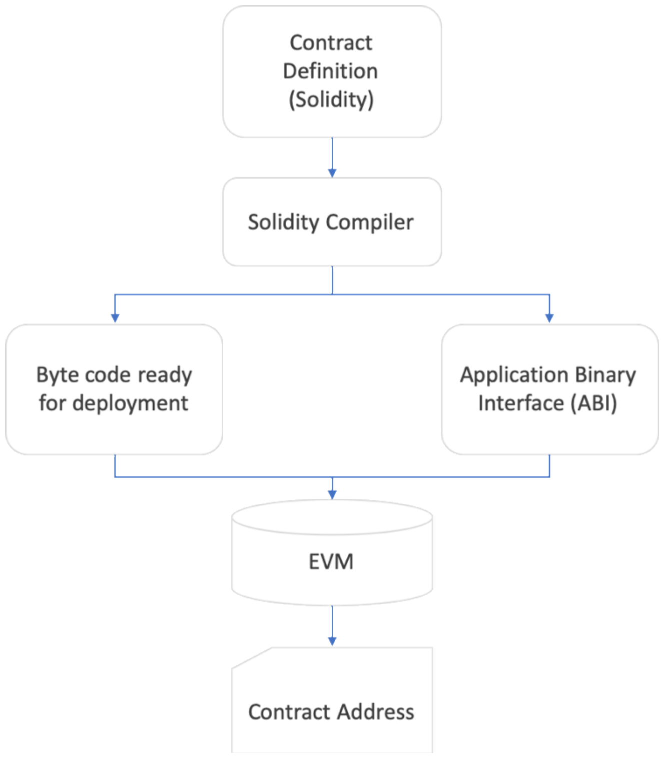

Contract Definition (Solidity)
Solidity Compiler
Byte code ready for deployment
Application Binary Interface (ABI)
EVM
Contract Address

Figure 11: Compiler operation flow

has actually been deployed on the Ethereum blockchain [47].

Remix is a powerful tool that supports a built-in compiler (Figure 12). It is necessary to ensure that the Remix compiler matches the Pragma Solidity version used in the contract. Pragmas are common instructions for compilers about how to treat the source code (e.g. pragma once) [48].

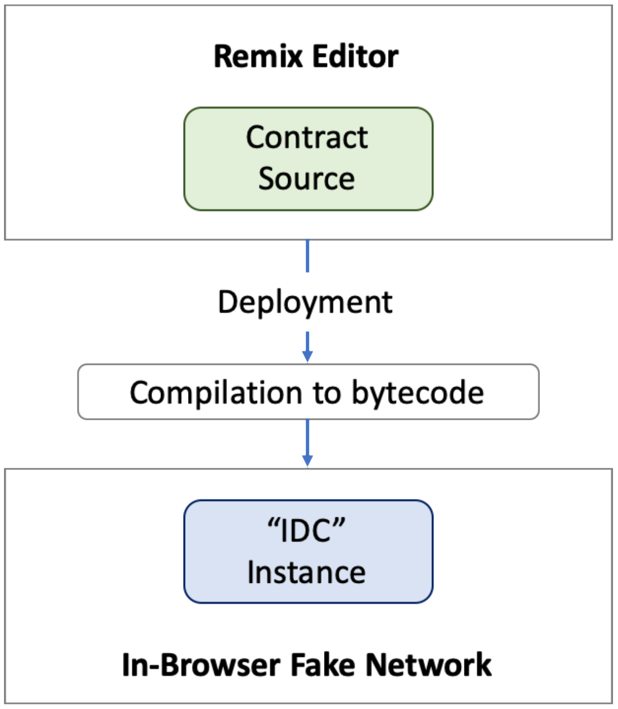


Figure 12: Remix compiler

The other vital part of developing a smart contract is the deployment (chapter 4.1) to an Ethereum network. In Remix, there are three options to deploy a contract: by using a JavaScript Virtual Machine (JVSM), through an injected Web3 option like Metamask, or with a web3 provider, where Remix will connect to an Ethereum Client – or node – by providing the corresponding URL. Metamask and Web3 packages will be explained next [REM creating a contract [48].

When deploying through the JSVM, Remix will create a virtual blockchain sandbox in the browser. Anytime the tab is reloaded, a "new blockchain" will be generated. Additionally, The

JSVM will generate five simulated account addresses to interact with the deployed contract (figure 13).

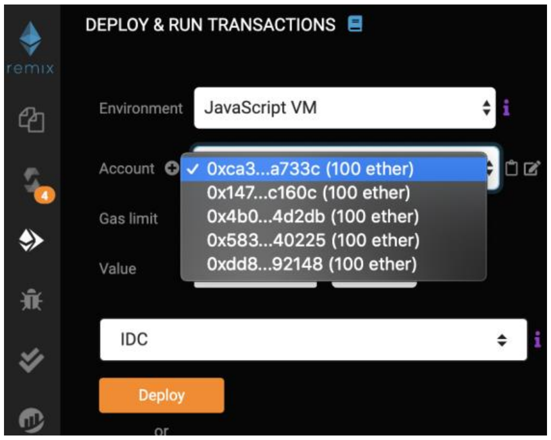

remix
DEPLOY & RUN TRANSACTIONS
Environment
JavaScript VM
Account
✓ 0xca3...a733c (100 ether)
0x147...c160c (100 ether)
0x4b0...4d2db (100 ether)
0x583...40225 (100 ether)
0xdd8...92148 (100 ether)
Gas limit
Value
IDC
Deploy
or

Figure 13: JSVM accounts display

When the gas Limit is set, the contract can be deployed. Next, the user will be able to interact with the deployed contract within the side panel where functions and other contracts can be called. The Terminal should confirm that the contract was successfully deployed also displaying the contract details.

3.2.2 Solidity

Solidity is an object-oriented programming language specifically invented for authoring smart contracts whilst interacting with an Ethereum Virtual Machine (EVM) [49]. Solidity is written into .sol files compatible with any standard code editor (Atom was used for coding of the IDC contracts). This language incorporates several libraries, inheritance and other features that facilitate the execution of smart contracts. Solidity is strongly typed unlike JavaScript which is dynamically typed. **[**50**]**

One of the main things to consider when coding in Solidity is that the language is in constant change. The version running as of the writing of the document is Solidity v0.5.11 [50].

The Solidity code predefined in a contract is not what is interacts with the Ethereum network; instead, the data is fed into a Compiler file. The Solidity Compiler[1] (solc) can be used through several ways, but the two most used ways are either through the Remix itself as

When not deployed through Remix, IDC smart contract was deployed using solc0.4.25 version[1] previously explained, and second, by installing a npm global instruction directly in the Terminal (*npm install -g solc)* [48].

As previously mentioned, if Remix is used, it is important to revise that the compiler version matches the Solidity (Pragma) version. However, If the contract is deployed using another alternative, a compiler script will be required. This script will be written in JavaScript syntax using standard library modules and will reference the Solidity file under the same directory.

The full Solidity documentation can be found in https://solidity.readthedocs.io/en/v0.4.25/

3.2.3 Ethereum Virtual Machine

The Ethereum Virtual Machine (EVM) is the infrastructure contained in Ethereum nodes that enable a runtime environment for smart contracts to run in the network. As DDoS attacks became widespread during recent years, the EVM focused on isolating the data deployed into the EVM. In this way, the EVM ensures that contracts have limited access to each other's state, entrusting the code deployed in the network [48].

As previously mentioned, Ethereum is supported by a built-in Turing program, meaning that the system enables a program or contract to find a solution for a certain problem albeit guaranteeing any performance regarding runtime and memory. Since the contracts' performance is constrained by a gas limit, the EVM is consider semi Turing [48].

Moreover, the binary data contained in a transaction is taken to be EVM bytecode and eventually executed in the blockchain. The output data of this execution is permanently stored as the code of the contract.

3.2.4 Metamask

Metamask is a browser extension commonly supported by Google Chrome that allow users to interact with dapps in regular web browsers; i.e. it is a link between an Ethereum node and a web browser [51]. Consequently, Metamask allow users to create accounts that can be linked to the main Ethereum Network, Ethereum Test Networks or any other customized RPC (Remote Procedure Call). Any individual with a Metamask account will be able to perform real transactions, test developed code using free Ether or even activate a local host (8545) to host an EVM.

To generate a Metamask account, instructions can be found on https://metamask.io/ After installing the extension and entering the corresponding password, an account will be created (figure 14). Each account created will have a public address, a public key and a private key. The address is a unique identifier that can be shared with any individual. It can be thought as a user name. The public and private key are pieces of information that will act as security protocols as previously revised in the last chapter. These three elements will be stored as hexadecimal values.

In addition, to enhance Metamask's security, when an account is created, a 12 word "password" called mnemonic phrase will be generated. Mnemonics are comprised by easy to remember words. One of the most important faculties of mnemonic phrases is the capability of restoring an address, public and private keys to a linked account. This is critical, especially when an account is storing real Ethers.

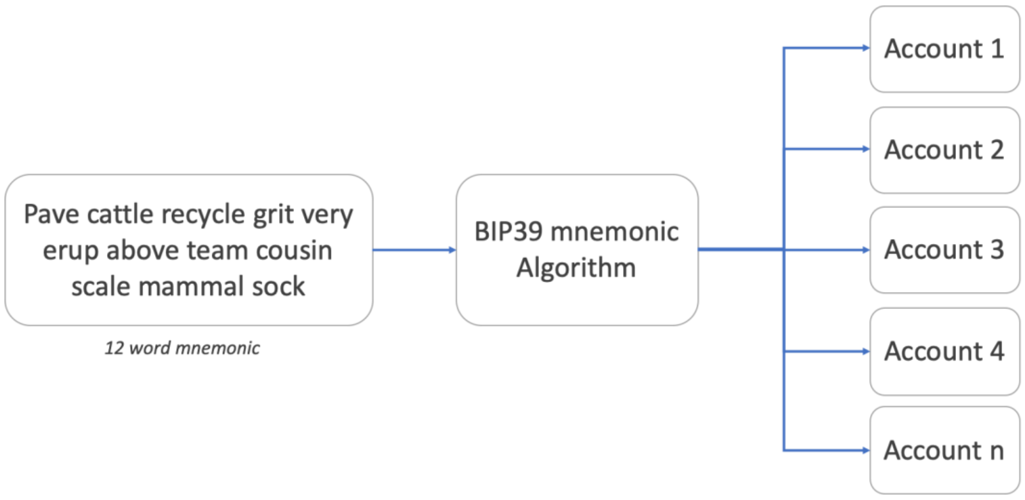


Figure 14: Metamask mnemonic architecture

Furthermore, when considering token valuation in a Metamask accounts, it is important to differentiate that Ethers owned in different test networks will have different values among each other network (figure 15). Each Metamask account will contain the same data for address, public and private key but coexisting in different networks. Once the account is created, Metamask's front displays all the available network options with the corresponding balance for each account, either real or test Ethers accordingly.

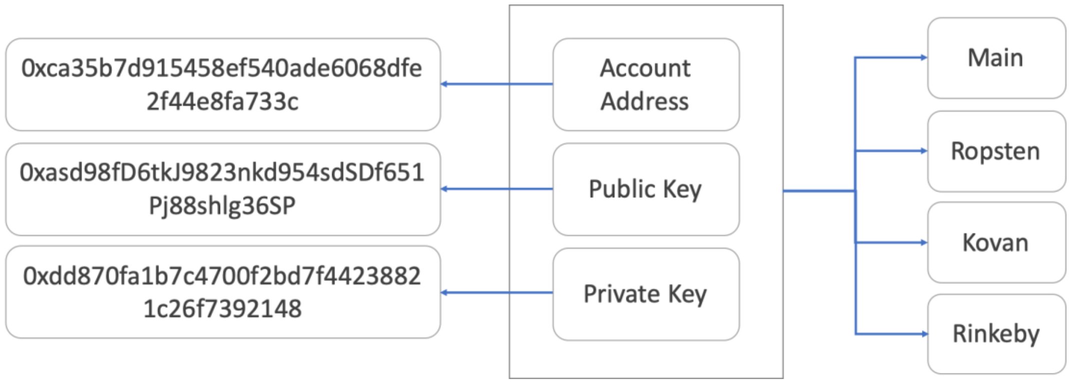


Figure 15: Metamask attributes coexistent environment

3.2.5 Infura

Transactions signed by a Metamask account need to be broadcasted to an EVM. Rather than running a full Ethereum node on a machine, there is a service that acts as a portal beyond Web3 into an Ethereum Public Network called Infura.

Infura works as an API for decentralized applications that grant users the ability to access an Ethereum Client without actually hosting a full node. The service hosted by Consensys [52] is a collection of full nodes on the Ethereum network that enable developers to connect to these nodes through its interface. As such, a significant amount of dapps' transactions run through Infura.

3.2.6 Web3.js

Web3 is used as the absolute end solution for establishing communications between a JavaScript app and the Ethereum network. Web3 can be best understood when compared to a sort of portal into the Ethereum network, it is a channel to enable program and programmatic access to an Ethereum network. Web3.js is a group of libraries that allows a user to interact with an Ethereum node through a HTTP connection[web3.js]. Web3.js can be installed by writing a npm global instruction directly in the Terminal[2] (*npm install -web3)* [53].

[2] Web3 version used for IDC contracts is Web3 v1.0.0 beta 35

This library will enable different functions as creating or deploying contracts, storing data and even performing currency transfers.

3.2.7 Truffle Framework

The Truffle framework is a series of open source software comprising three tools, Drizzle, Ganache and Truffle, that facilitate the creation and development of smart contracts and dapps. Each tool possesses different capacities depending on the user requirements. is library will enable different functions as creating or deploying contracts, storing data and even performing currency transfers. For the development of IDC contracts, Ganache and Truffle were used.

Drizzle is a collection of front-end libraries that mainly, simplify the interaction between a contract and an Ethereum node [54].

Ganache is used to create an Ethereum blockchain that runs locally. This tool is available as a desktop application or as a command-line tool (also known as TestRPC). One of the main features of Ganache is the creation of 10 unlocked accounts containing an address, a private key and 100 ETH each [55]. All 10 accounts are also linked to a mnemonic phrase. In this way, Ganache can be used to deploy contracts and run tests, as shown in 5.1. Another magnificent feature is that Ganache can serve as the Web3 Provider in Remix, so the 10 accounts that were automatically generated, can be used to deploy a contract on an Ethereum Test Network.

Ganache can also be linked to Metamask, as well as other additional features that require a deeper understanding of blockchain development such as setting the mining time of each block.

Finally, Truffle [56] is a very complete development environment used for the development of smart contracts seizing its compiling and deploying capabilities, as well as a robust testing framework and accessing Ethereum nodes (figure 16).

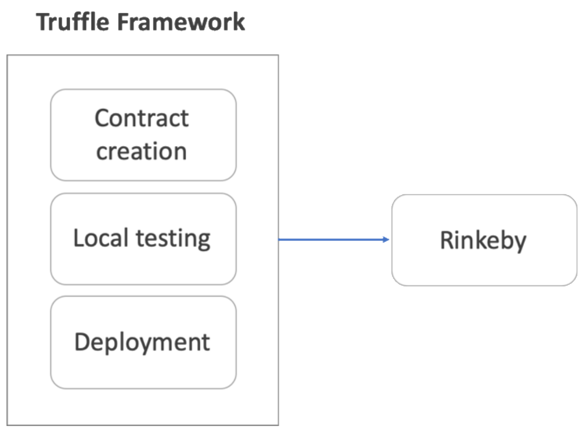


Figure 16: Truffle framework

3.2.8 Etherscan

When a transaction is successfully deployed on a public Ethereum Network, it creates a series of data related to its deployment such as the transaction hash, block, timestamp, value, etc.

Etherscan is a search engine that allows any individual, first to confirm if a transaction was properly deployed, to look for the transactional data of a deployed transaction. Instances can be searched by contract address, transaction hash or block number.

3.3 Specific requirements

In addition to the aforementioned requirements, in order to properly either use or develop a contract, further requisites are needed for a proper function.

- Install the Node Package Manager (NPM) required to install additional functions and libraries
- In the Terminal, using the npm global functions, install *solc, web3, ganache-cli and truffle hd wallet provider*
- Have a web browser that supports the current version of Remix and Metamask. These browsers include Google Chrome, Mozilla Firefox, Opera and Brave
- Create a Metamask account ensuring to keep safe the mnemonic phrase
- Request free Ether on the Rinkeby Faucet https://faucet.rinkeby.io/
- Create an Infura project to link with the contract
- Recommended to download Ganache console software to interact with deployed contracts

4. Implementation

This chapter will cover first, the most utilized approaches to implement or deploy a smart contract in an Ethereum public network are briefly revised, as well as how transactions work in the Ethereum context; afterwards, the application case used to apply the IDC smart contracts is explained; and finally, a high-level approach of the IDC smart contracts architecture is detailed.

4.1 Deploying a Smart Contract

There are several ways to deploy a Smart Contract. The network where a contract wants to be deployed plays the first role on choosing the appropriate method to deploy it, whether it's a local host, a Test Network or the Ethereum Main Network [57]. Then, depending on the complexity of each contract, there are several methods that allow the user to have different levels of control and interaction. For example, if a contract includes convoluted functions, it may be better to host an Ethereum Client locally via Geth or Parity (figure 17).

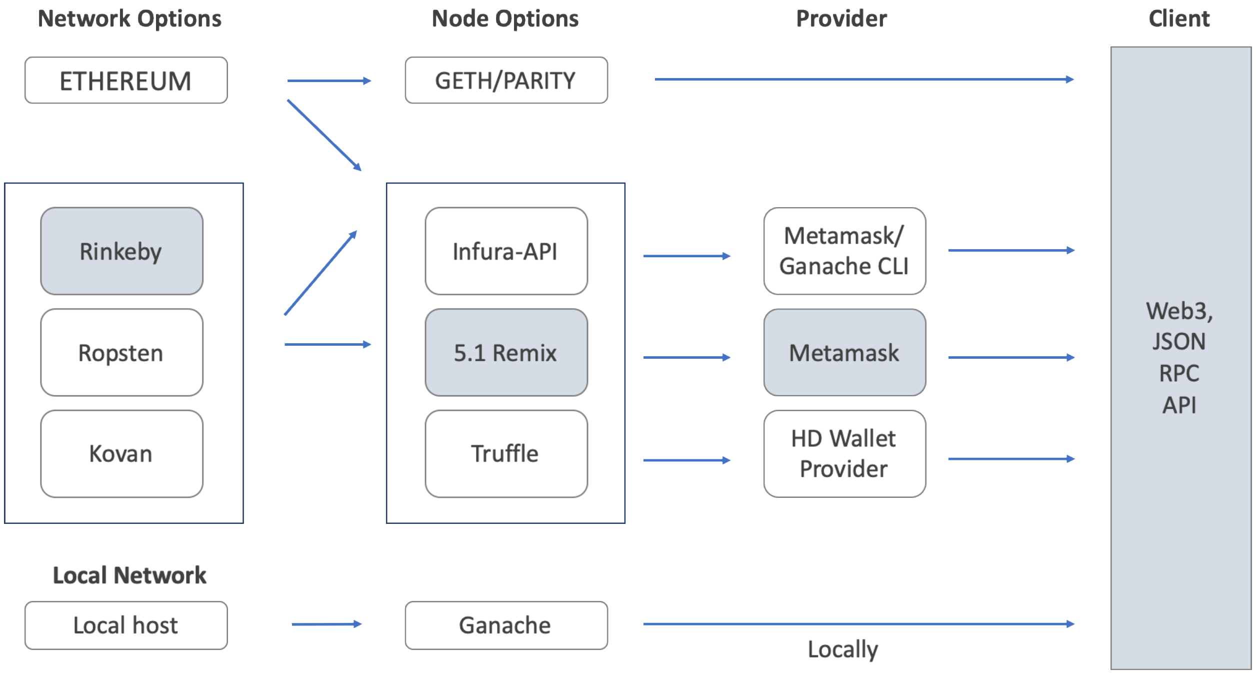


Figure 17: Smart Contract Deployment Overview Source (adapted from [57])

The IDC contracts deployed to the Rinkeby network using Remix. As explained in section 3.2.1, Remix offers a wide array of options at the moment of deploying a contract. In case of the IDC contracts, the injected Web3 option was chosen. With this method, Remix will connect to a Metamask account, for example, an account linked to the Rinkeby network (figure 18). The detailed process will be covered in section 5.

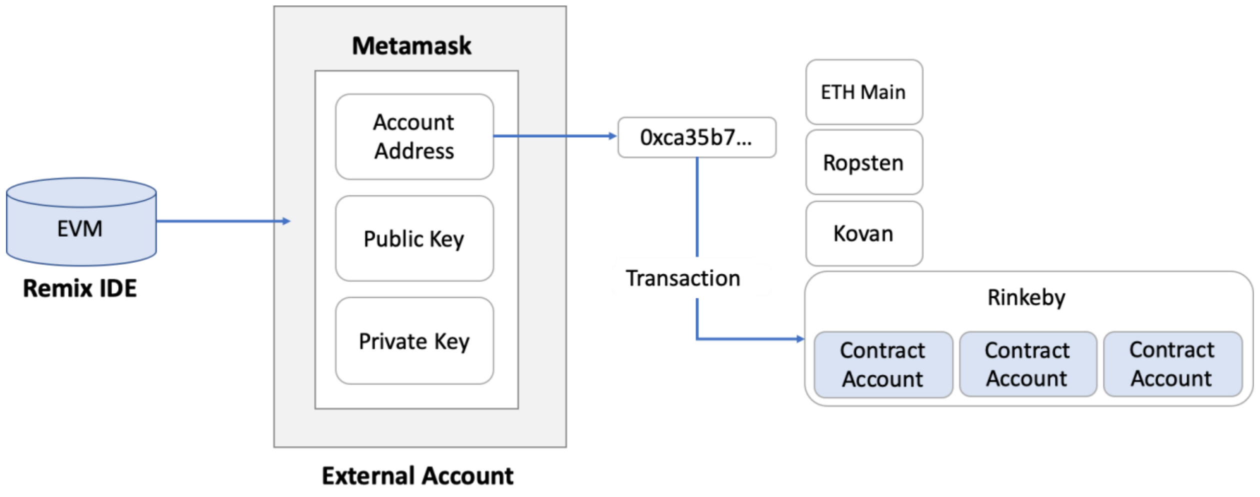


Figure 18: Contract deployment flow using Remix via Metamask

4.2 Transactions

No matter the way in which a contract was deployed it is important to differentiate the type of actions a contract can perform.

A transaction object is a cryptographically-signed record that describes the attempt of one account to interact with another account. These transactions are publicly recorded on the blockchain by taking the form of an object in programming languages such as Solidity. As Wood [58] indicates, users create transactions to the Ethereum network in order to: create new contracts; invoke functions of a contract or transfer between accounts.

There are two main types of transactions: get functions and set functions. Whereas the first is a single call to a specific function or contract, the latter involve direct relation with the blockchain changing the content on it. When a transaction is broadcasted and published into the network, miners will consume Ether for writing an operation that will affect related accounts whilst updating the blockchain. Therefore, each set function will have a gas cost as stated before. On the other hand, get functions can be called without incurring in any gas charge.

No matter the type of transaction, it will always contain a group of properties as shown in table 5.

Transaction Properties	
Name	Description
nonce	How many times a sender has sent a transaction. Different from the nonce in a block
to	Address of the message recipient
value	Number of Ether or Wei to be transfer to the recipient
gasPrice	Amount of Wei the sender is willing to pay per gas unit to get the transaction mined
gasLimit	Units of gas this transaction can consume
v	Cryptographic pieces of data generated from the sender's private key to determine the address of origin
r	
s	

Table 5: Transaction properties

4.3 Application case: Gluten Free Oats

IDC contracts aim to generate a traceable journey of a given product or service along the supply chain to bring **transparency** and **trust** through a **tamper-proof system** based on blockchain technologies. To gain a more accurate impression of the potential performance of the IDC contract, a real-life production process was studied; in this case, the production process of a Gluten-Free Oat Production Center (GFO Farm) was used as a parameter to test the application.

The GFO Farm crops and produces a specific type of oat grain that require special treatments at different times of the production process. Under this context, IDC would be an ideal fit to ensure the cultivators that the grains are treated adequately in specific stages whilst also "certifying" a product from the customer standpoint. Additionally, a tracking process can be incorporated between different stages of the production if required.

4.3.1 High-level process

1. Field cultivations: although the Gluten-Free qualification is not earned at the initial stage of the process, it is important to acknowledge the provenance and amount of the seed, as well as other specifications relevant for the cultivation stage.
    a. Sowing – seed is certified depending upon how many times it has been used i.e. Seed that comes direct from the breeder is free from any contamination, later generations many become more contaminated but there are standards for each generation.
    b. Previous cropping – important as grain lost in the field can appear in following crops.
    c. Field cultivations – verification of seed origin and amount cultivated.
    d. Other inputs such as agro-chemicals and fertilizer - there are restrictions on the plant growth stages that certain agro-chemicals can be used on the crop.
2. Harvest and on-far storage
    a. Many combine harvesters now have yield monitors which shows the volume of grain coming off different areas of each field through a combination of GPS and automatic in-harvest sample weighing on the combine. This is usually displayed as a heatmap of the field.
    b. Transport from harvester to on-farm storage – usually done by the farm staff and equipment. Care needs to be taken to thoroughly clean down machinery and storage before and in use between different crops. Each trailer load is usually weighed before the crop is put into store.
3. Transport to processor – usually done by external haulage contractor. Again, thorough cleaning of the lorry necessary between loads.
4. Storage and processing. Most oats are processed in 'conventional' rather than dedicated gluten free facilities so need to be cleaned down between batches. Other competitor farms are building their own dedicated gluten-free processing facility on one of the farms.

a. After processing the oats will generally be packed into bags of different sizes depending upon the customer's requirements. Occasionally for large orders they may be transported in bulk but this is not that common.

b. They will be transported to the manufacturers either as pallets of small bags or in large (0.5 or 1 ton) bags.

5. Manufacturer – The main concern of most manufacturers is that the oats conform to the required specification. The gluten-free aspect could be taken on trust but there is likely to be some form of paper trail for this although it could well be disjointed and at present there is no gluten-free assurance / certification scheme for gluten–free oats.
6. Retailer – Generally will expect the specification / safety / certification of the product to be the responsibility of the manufacturer so it is taken on trust.
7. Consumer – generally expect the retailer to provide them with safe food of a consistent quality.

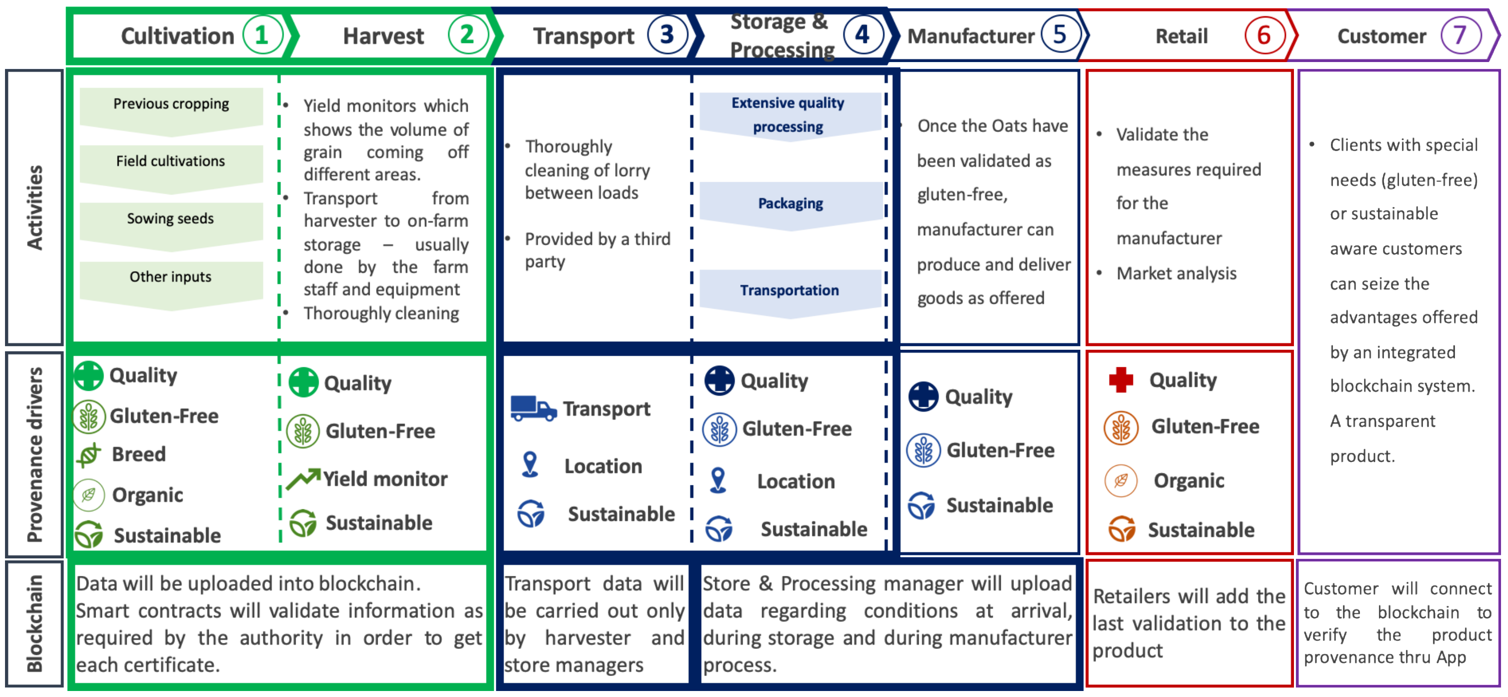

Cultivation 1
Harvest 2
Transport 3
Storage & Processing 4
Manufacturer 5
Retail 6
Customer 7
Activities
Previous cropping
Field cultivations
Sowing seeds
Other inputs
Yield monitors which shows the volume of grain coming off different areas.
Transport from harvester to on-farm storage – usually done by the farm staff and equipment
Thoroughly cleaning
Thoroughly cleaning of lorry between loads
Provided by a third party
Extensive quality processing
Packaging
Transportation
Once the Oats have been validated as gluten-free, manufacturer can produce and deliver goods as offered
Validate the measures required for the manufacturer
Market analysis
Clients with special needs (gluten-free) or sustainable aware customers can seize the advantages offered by an integrated blockchain system. A transparent product.
Provenance drivers
Quality
Gluten-Free
Breed
Organic
Sustainable
Quality
Gluten-Free
Yield monitor
Sustainable
Transport
Location
Sustainable
Quality
Gluten-Free
Location
Sustainable
Quality
Gluten-Free
Sustainable
Quality
Gluten-Free
Organic
Sustainable
Blockchain
Data will be uploaded into blockchain. Smart contracts will validate information as required by the authority in order to get each certificate.
Transport data will be carried out only by harvester and store managers
Store & Processing manager will upload data regarding conditions at arrival, during storage and during manufacturer process.
Retailers will add the last validation to the product
Customer will connect to the blockchain to verify the product provenance thru App

Figure 19: Gluten-free Oats Production process / IDC scope

The main scope of the current IDC contracts is solely focused on the first two stages of the process, as well as the Transport and Storage phases. Albeit the IDC contract contains functionalities to trace a product's journey, advanced infrastructure is required to provide an efficient tracking system. Therefore, these stages are not considered currently covered by the contract's setup (figure 19).

4.3.2 Stakeholders

Smart contracts can be triggered by key stakeholders at critical stages of the process, acting as checkpoints. Having verifiable data on the blockchain can ease audit services or can serve to meet certain requirements (e.g. Scotch Beef requires farmers to be a member of the QMS Assurance Scheme).

Each stakeholder will be able to call a smart contract that will verify whether a condition is being met. Initially, retailers and manufacturers should cooperate to act as accreditors while the blockchain smoothly enables those capabilities.

In the other hand, the final customer should be able to see the provenance of a product through the web application front-end (interfaced to the blockchain). Many consumers are becoming increasingly interested in how their food is produced and its environmental impact so blockchain does present an opportunity to fulfill this condition.

The main stakeholders identified for the GFO Farm case were:

- Seed breeders
- Cultivation supervisor
- Machinery supervisor
- Harvest supervisor
- Transport & logistics
- Retailing
- Consumer service
- Legislation and certification
- Farming & food expert
- Environmental Health and Trading Standards

4.4 IDC Smart Contract: high-level design

The IDC contract was developed to comply with certain logic based on the GFO Farm process. The variables and functions were defined focusing on the aforementioned stages of the process.

The contract is called by the Administrator figure. First, a producer must be registered for eventually recording a seed, crop or harvest. This marks the first stage of the contract logic where duplicated or null values will be marked as errors and thus, reverted by the EVM. Prior to the cultivation stage, a breeder should register the seed data (figure 20) in the blockchain. Again, some requirements such as null values, and a registered producer will cause an EVM revert if occurring.

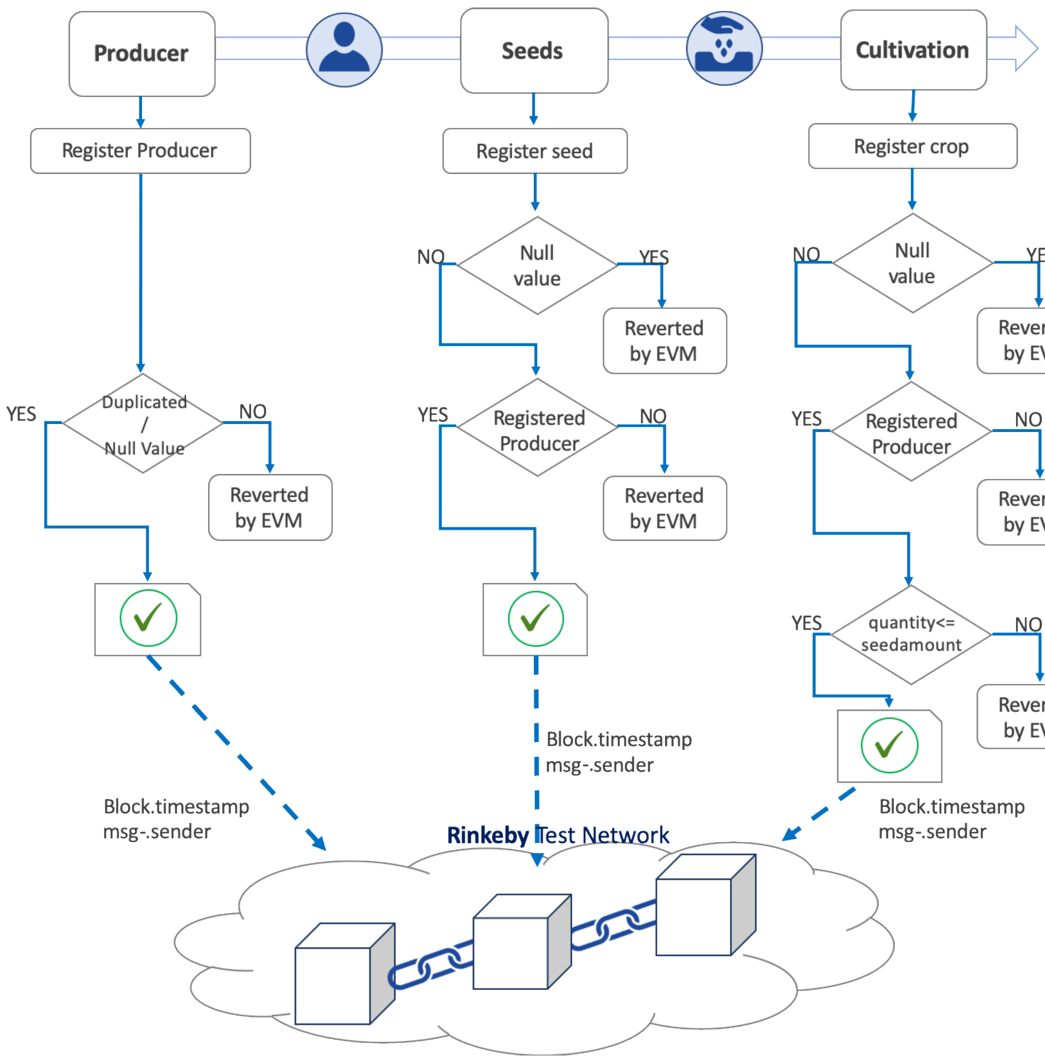


Figure 20: Smart Contract – Cultivation logic

For the Cultivation stage, a farmer will register a crop (figure 20). In this stage, data such as agro-chemicals, previous cropping and sowings will be registered. Also, if the quantity registered for a crop exceeds the amount of a certain registered seed, an error will be thrown. This will enhance the traceability by ensuring that the seeds used for each crop can be identified avoiding mixtures.

Then, a registered producer will record the harvest data (figure 21). The logic for this part of the process is similar than the one used for Cultivation; however, since the oat at this point is a different good, the quantity will only be registered but not monitored.

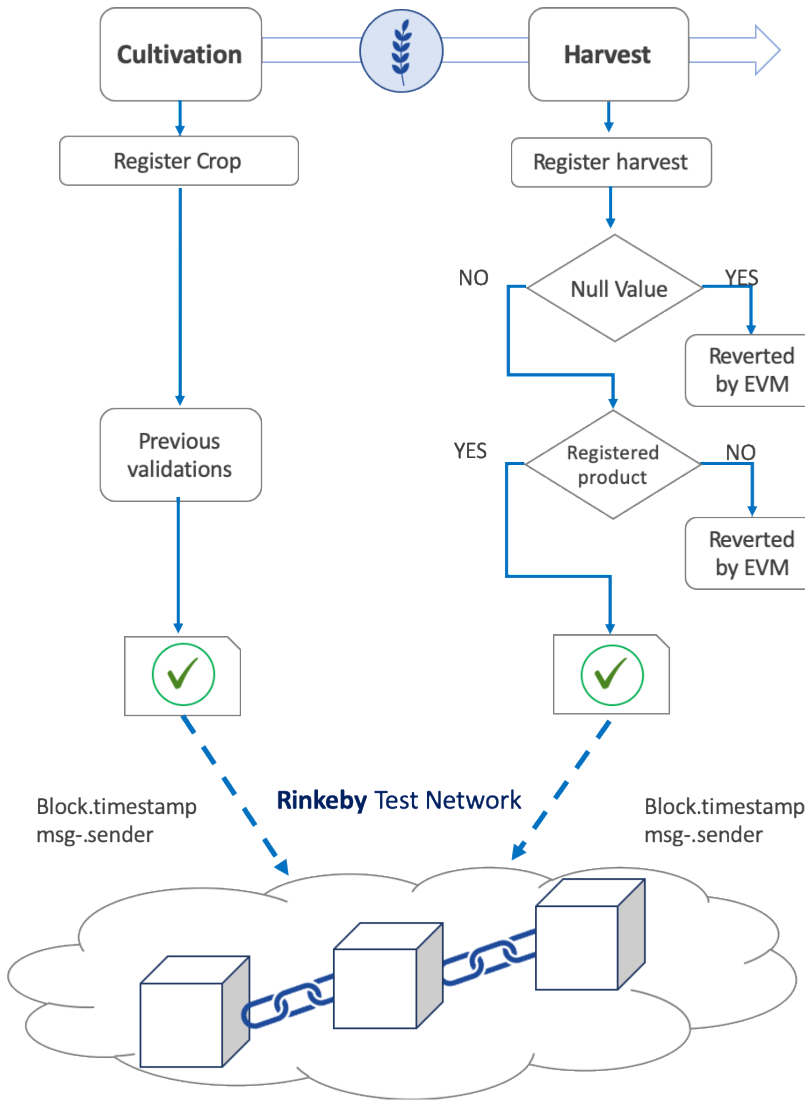


Figure 21: Smart Contract – Harvest logic

Finally, a product can be tracked from the time of the harvest collection to the manufacturing & storage phase. In this part, the contract registers a shipment of a harvest from a registered producer. Afterwards, on the other end, a party in charge of receiving the product, will verify its state and will again verify that the amount received complies with requirements upon agreement (figure 22).

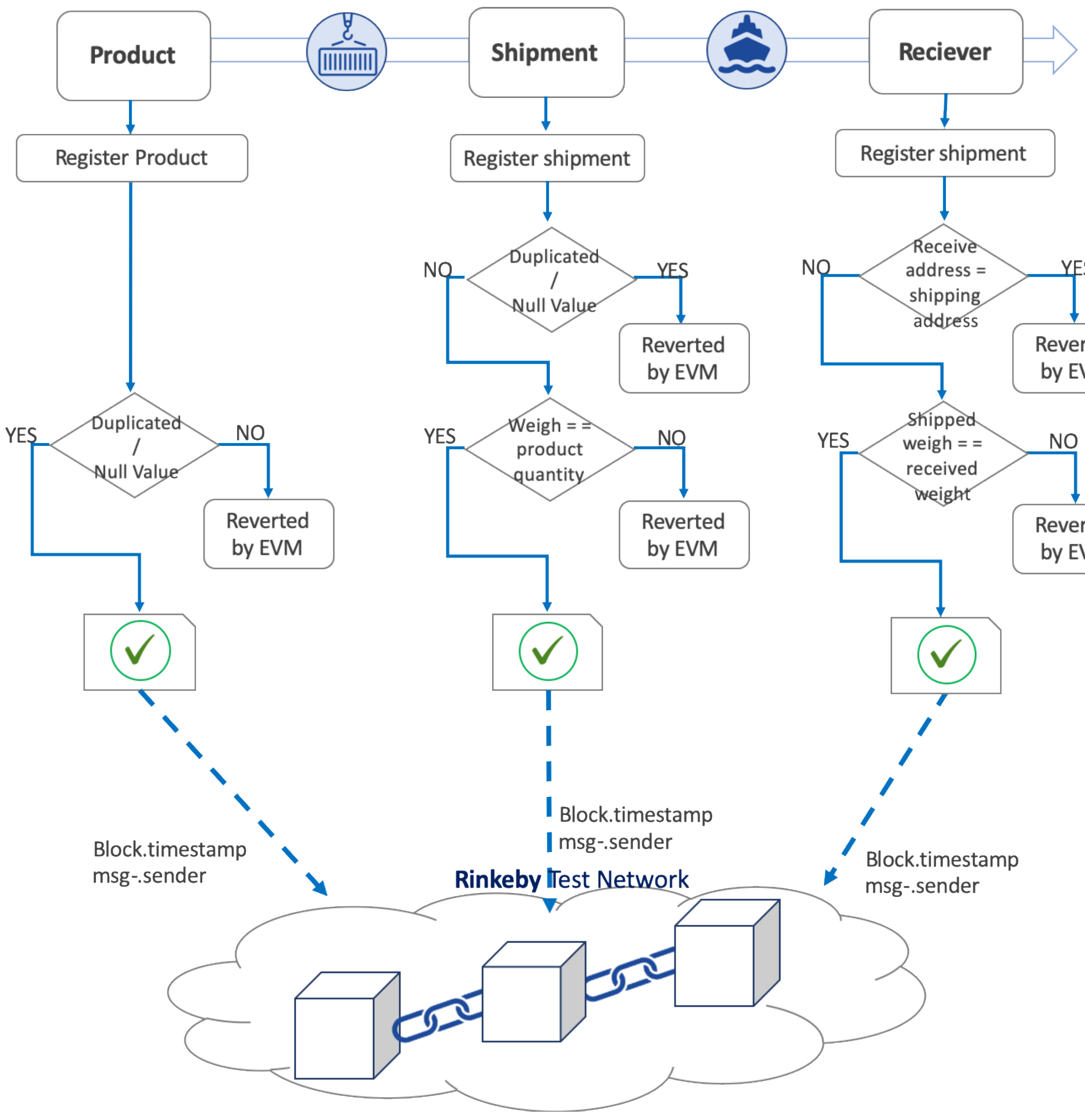


Figure 22: Smart Contract - Tracking logic

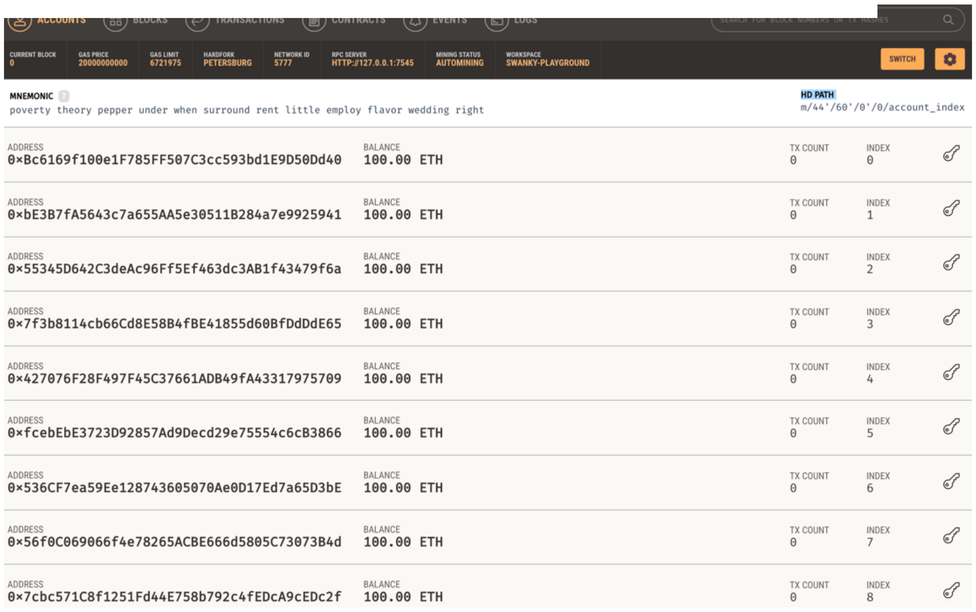
The EVM will again revert the transaction in case the receiver has the same address as the sender. Also, there is a weight verification.

5. Functionalities

In this section, the whole contract functionalities are displayed. The setup procedures required for the deployment are also established. The contract boilerplate (tables A-D) enlists the variables used, their type and description. Also, the list of functions as well as the functionality are covered. (Appendix).

5.1 Metamask and Remix setup

As explained in 4.1. Remix can deploy a contract to an Ethereum Test Network with an injected Web3 via Metamask. Before connecting the IDE to Metamask, seven accounts were imported from a local Ganache project. As a reminder, Ganache serve as a local blockchain creating 10 open accounts every time this framework is run.



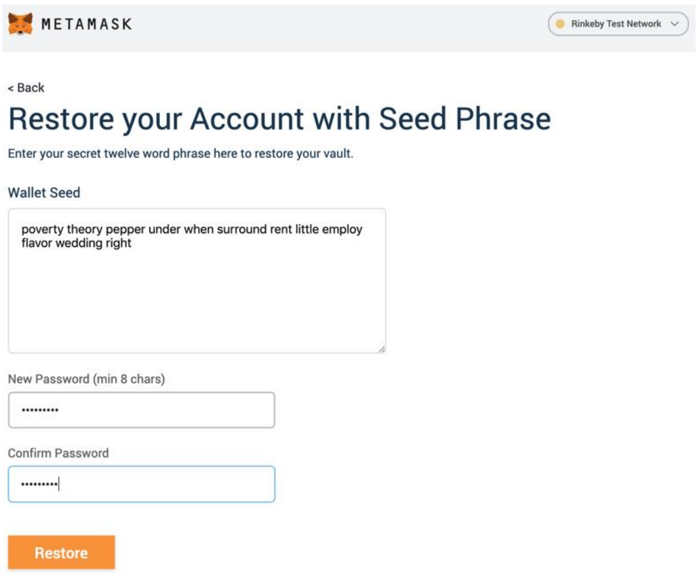
CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
0	20000000000	6721975	PETERSBURG	5777	HTTP://127.0.0.1:7545	AUTOMINING	SWANKY-PLAYGROUND

MNEMONIC
poverty theory pepper under when surround rent little employ flavor wedding right

HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×Bc6169f100e1F785FF507C3cc593bd1E9D50Dd40	100.00 ETH	0	0
0×bE3B7fA5643c7a655AA5e30511B284a7e9925941	100.00 ETH	0	1
0×55345D642C3deAc96Ff5Ef463dc3AB1f43479f6a	100.00 ETH	0	2
0×7f3b8114cb66Cd8E58B4fBE41855d60BfDdDdE65	100.00 ETH	0	3
0×427076F28F497F45C37661ADB49fA43317975709	100.00 ETH	0	4
0×fcebEbE3723D92857Ad9Decd29e75554c6cB3866	100.00 ETH	0	5
0×536CF7ea59Ee128743605070Ae0D17Ed7a65D3bE	100.00 ETH	0	6
0×56f0C069066f4e78265ACBE666d5805C73073B4d	100.00 ETH	0	7
0×7cbc571C8f1251Fd44E758b792c4fEDcA9cEDc2f	100.00 ETH	0	8

Next, it is necessary to set up Metamask by importing those Ganache accounts into the browser extension. This can be done by restoring a Metamask account using the mnemonic or seed phrase



Restore your Account with Seed Phrase

Enter your secret twelve word phrase here to restore your vault.

Wallet Seed

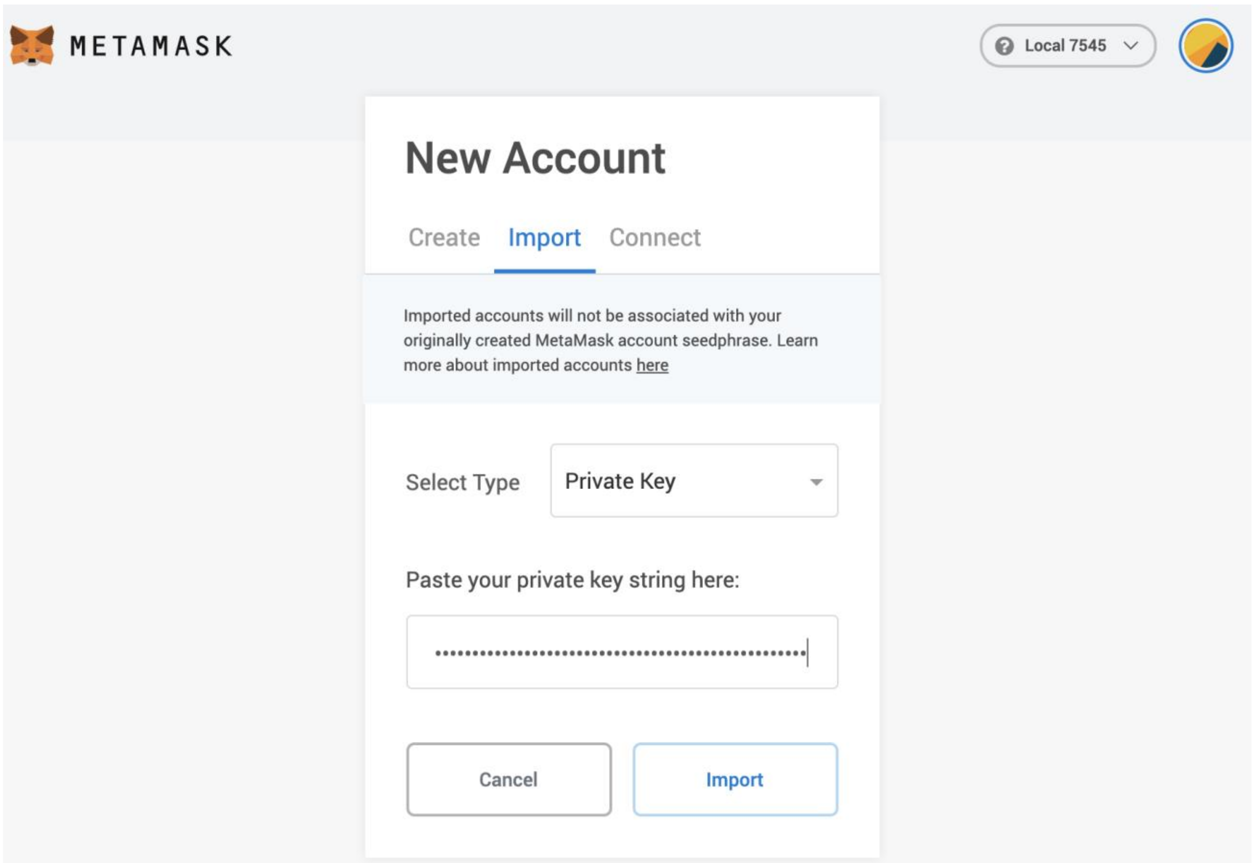
poverty theory pepper under when surround rent little employ flavor wedding right

New Password (min 8 chars)

Confirm Password

This will create a new account which, in this case, will be connected to the Rinkeby network.

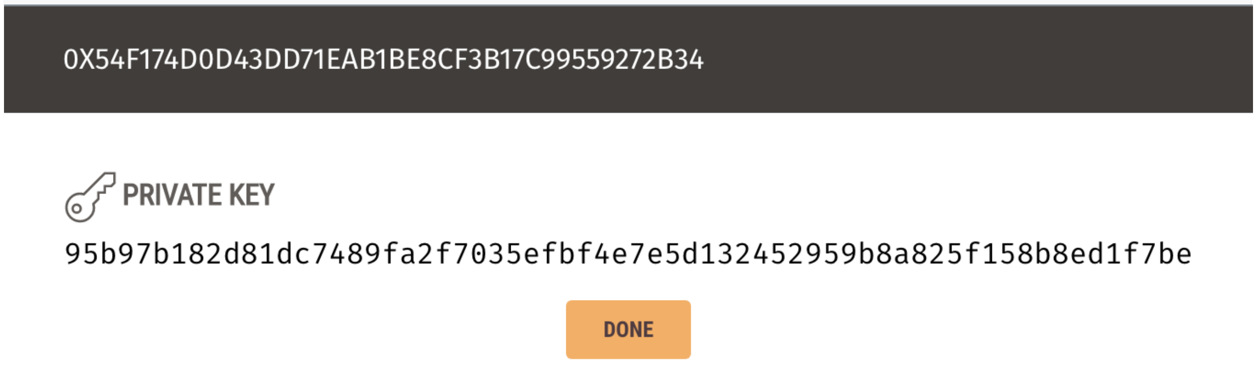
The rest of the Ganache accounts can be imported using the Private Key



Imported accounts will not be associated with your originally created MetaMask account seedphrase. Learn more about imported accounts here

Select Type: Private Key

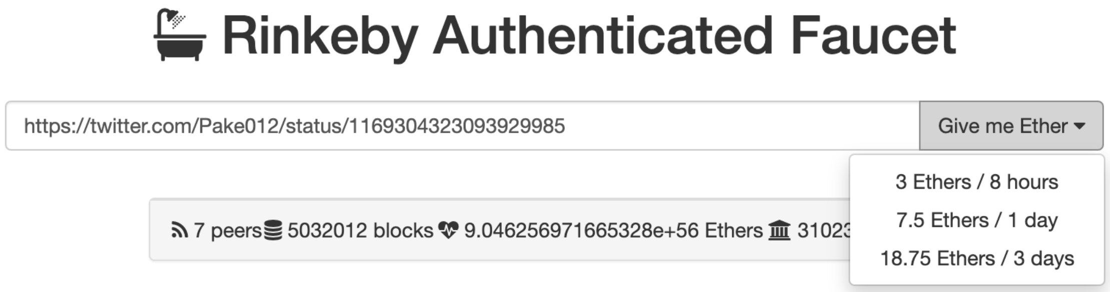
Paste your private key string here:


0X54F174D0D43DD71EAB1BE8CF3B17C99559272B34

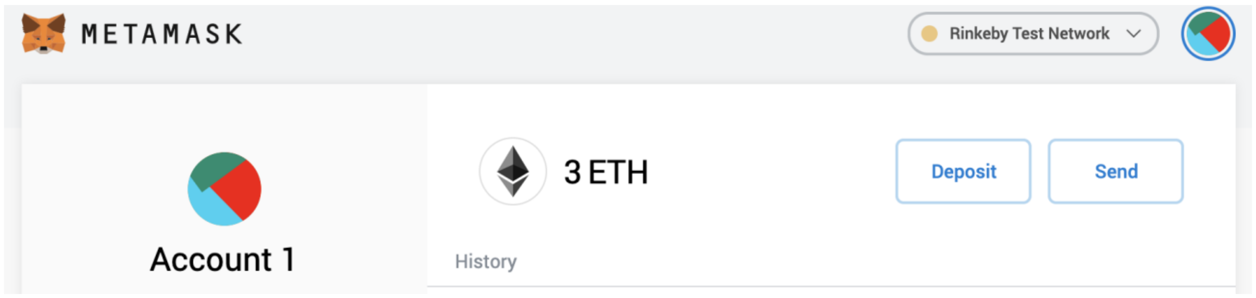
PRIVATE KEY

```
95b97b182d81dc7489fa2f7035efbf4e7e5d132452959b8a825f158b8ed1f7be
```

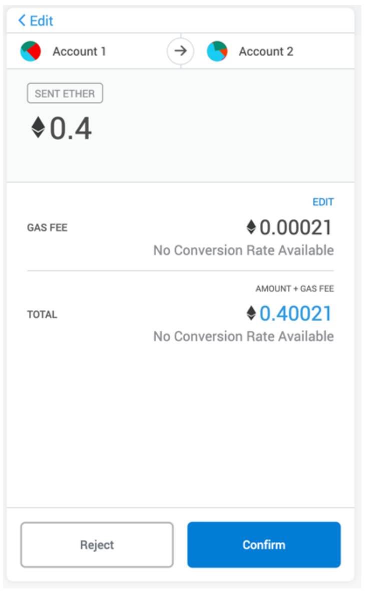
Unfortunately, the Ethers contained in the Ganache accounts can only be used locally, so when connected to the Rinkeby, these are lost. However, as mentioned in section 3.3, through the Rinkeby faucet, Rinkeby Ether can be requested following the procedure specified on the site.

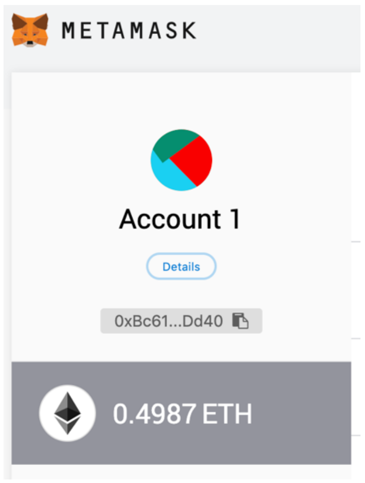

Rinkeby Authenticated Faucet
https://twitter.com/Pake012/status/1169304323093929985
Give me Ether
3 Ethers / 8 hours
7.5 Ethers / 1 day
18.75 Ethers / 3 days
7 peers 5032012 blocks 9.046256971665328e+56 Ethers 31023

For this simulation, 3 Ethers were requested, which is more than enough to fully test the contract. When the procedure is completed, and after a 30 second wait approximately, the Metamask account should reflect the new balance.

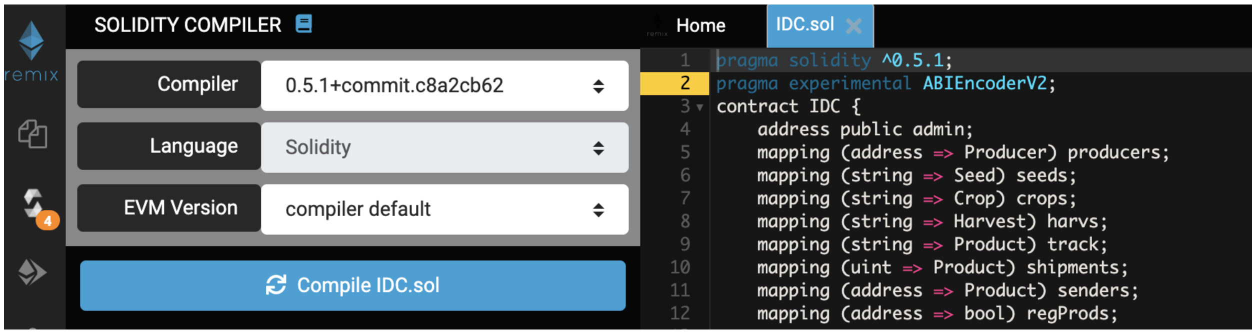


Then, from Account 1, the rest of the accounts are credited with 0.4 ETH. Now, all accounts have sufficient funds to interact with the contract. After funding six additional accounts with 0.4 ETH each plus the transfer fees, the balance in Account 1 is 0.5ETH approximately.





Then, start Remix (https://remix.ethereum.org/). As previously mentioned, it is important to set the compiler in the same versioning as the writing of the contract.



Compiler: 0.5.1+commit.c8a2cb62
Language: Solidity
EVM Version: compiler default

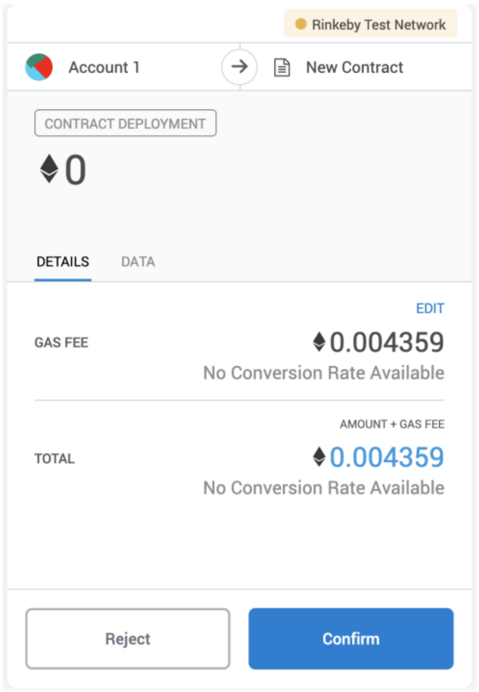
```
pragma solidity ^0.5.1;
pragma experimental ABIEncoderV2;
contract IDC {
    address public admin;
    mapping (address => Producer) producers;
    mapping (string => Seed) seeds;
    mapping (string => Crop) crops;
    mapping (string => Harvest) harvs;
    mapping (string => Product) track;
    mapping (uint => Product) shipments;
    mapping (address => Product) senders;
    mapping (address => bool) regProds;
```

Also, set the environment at Injected Web3. By accessing Web 3, all the Metamask accounts will be automatically connected.

5.2 IDC deployment

Constructor Function

The contract will be deployed from Account 1, so it will be deemed as Administrator *(admin).* A contract is deployed by executing the constructor function which will create an instance of the contract to interact with. A constructor function is considered a set function, hence, a confirmation of the gas fee required will be sent.


Rinkeby Test Network

Account 1 → New Contract

CONTRACT DEPLOYMENT

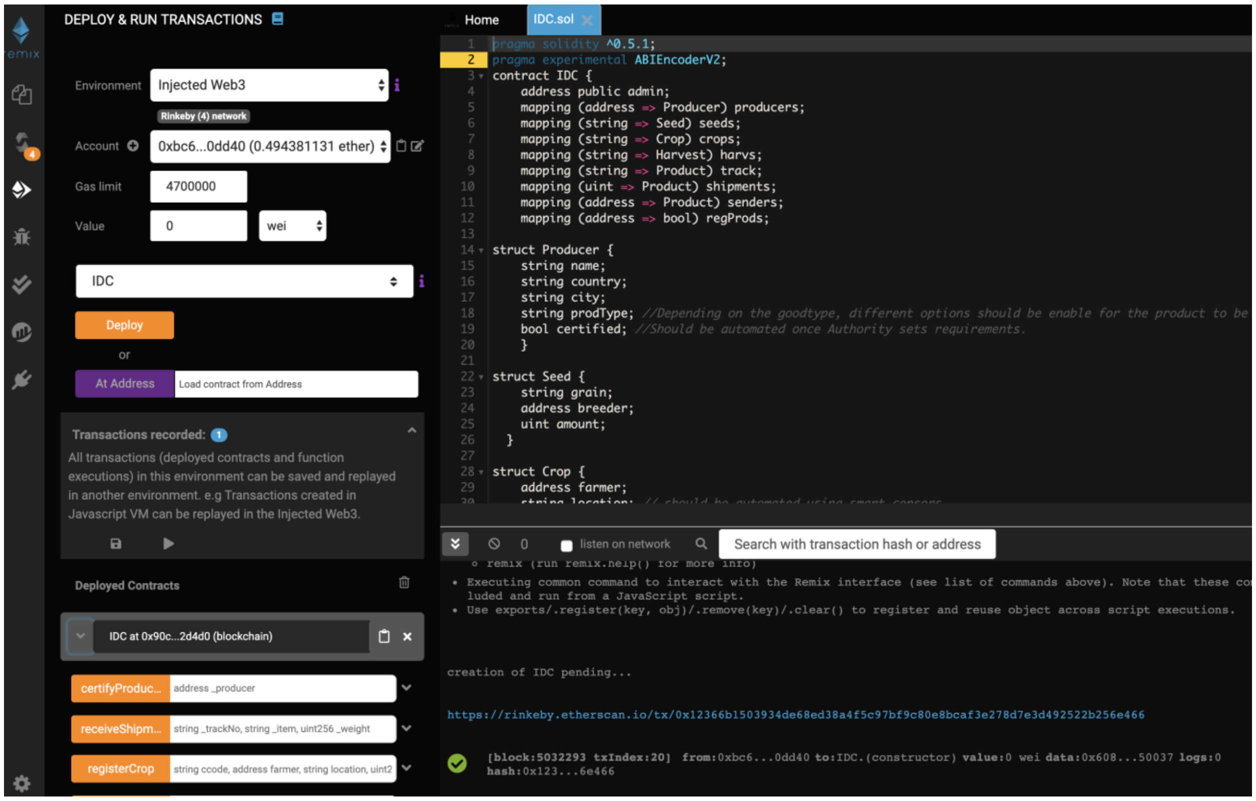
0

DETAILS DATA

GAS FEE	EDIT 0.004359 No Conversion Rate Available
TOTAL	AMOUNT + GAS FEE 0.004359 No Conversion Rate Available

Reject Confirm

When the contract is deployed, the get and set functions, as well as the defined variables will appear on the Remix side panel. The transaction receipt will also be displayed in the console.


DEPLOY & RUN TRANSACTIONS

Environment	Injected Web3 (Rinkeby (4) network)
Account	0xbc6...0dd40 (0.494381131 ether)
Gas limit	4700000
Value	0 wei

IDC

Deploy

or

At Address — Load contract from Address

Transactions recorded: 1

All transactions (deployed contracts and function executions) in this environment can be saved and replayed in another environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.

Deployed Contracts

IDC at 0x90c...2d4d0 (blockchain)

certifyProduc...	address _producer
receiveShipm...	string _trackNo, string _item, uint256 _weight
registerCrop	string ccode, address farmer, string location, uint2

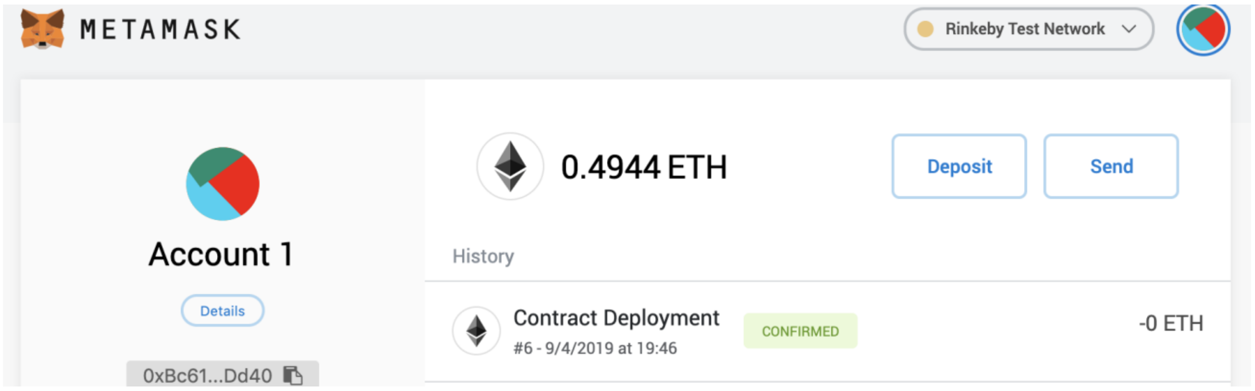
Home IDC.sol

```
pragma solidity ^0.5.1;
pragma experimental ABIEncoderV2;
contract IDC {
    address public admin;
    mapping (address => Producer) producers;
    mapping (string => Seed) seeds;
    mapping (string => Crop) crops;
    mapping (string => Harvest) harvs;
    mapping (string => Product) track;
    mapping (uint => Product) shipments;
    mapping (address => Product) senders;
    mapping (address => bool) regProds;

struct Producer {
    string name;
    string country;
    string city;
    string prodType; //Depending on the goodtype, different options should be enable for the product to be
    bool certified; //Should be automated once Authority sets requirements.
    }

struct Seed {
    string grain;
    address breeder;
    uint amount;
  }

struct Crop {
    address farmer;
```

0 listen on network — Search with transaction hash or address

```
  o remix (run remix.help() for more info)
• Executing common command to interact with the Remix interface (see list of commands above). Note that these co
  luded and run from a JavaScript script.
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.


creation of IDC pending...

https://rinkeby.etherscan.io/tx/0x12366b1503934de68ed38a4f5c97bf9c80e8bcaf3e278d7e3d492522b256e466

✓ [block:5032293 txIndex:20] from:0xbc6...0dd40 to:IDC.(constructor) value:0 wei data:0x608...50037 logs:0
  hash:0x123...6e466
```

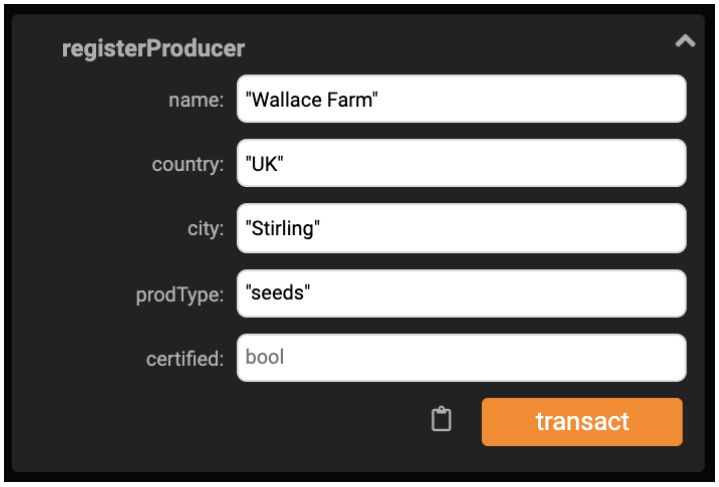
On the other hand, the Metamask account will reflect the transaction as well as the new balance after it.



Contract Deployment CONFIRMED -0 ETH
#6 - 9/4/2019 at 19:46

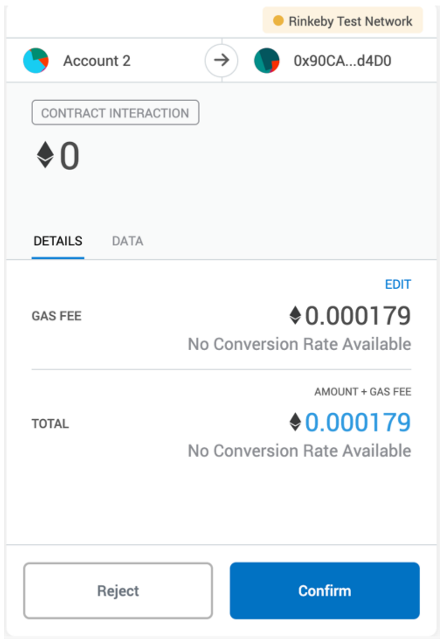
Register Producer

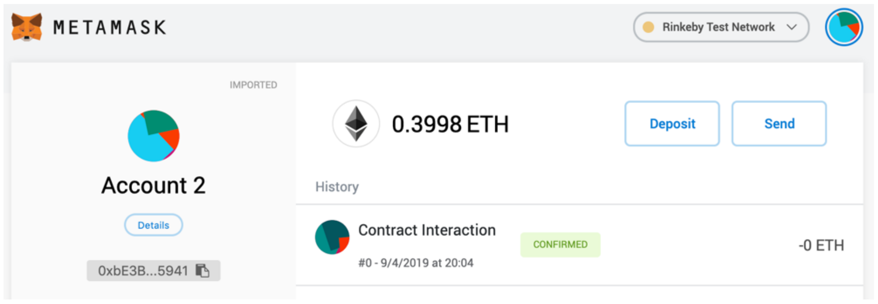
This function registers the data for a Producer and stores it on the blockchain. In this way, an account is recorded as a registered producer unlocking the upcoming contract functions. Addresses corresponding to accounts 2 to 6 will call the function and hence be registered as producers. Restrictions prevent storing a null or duplicated value.



name:	"Wallace Farm"
country:	"UK"
city:	"Stirling"
prodType:	"seeds"
certified:	bool

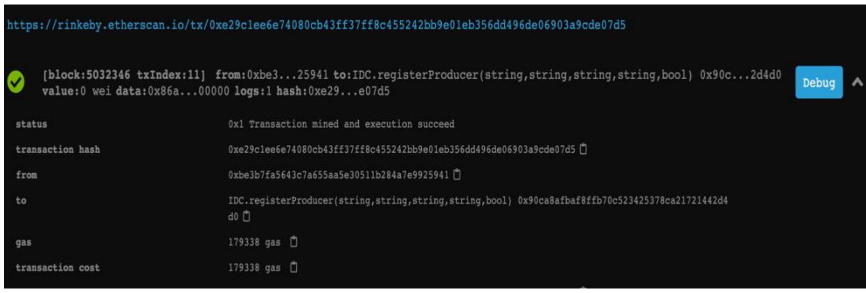
When the function is transacted, a confirmation from Metamask will be required. Once the transaction is mined, it will be reflected on the account, as well as in the console. Additionally, the data can be monitored as an event through Etherscan.



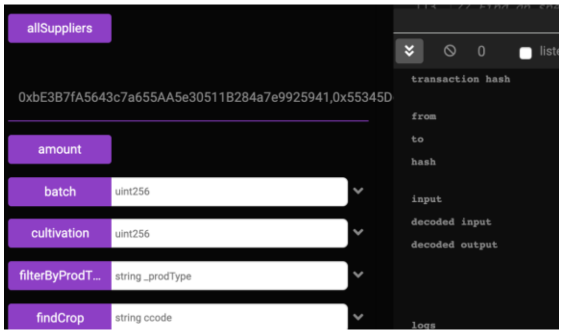


All Suppliers

This is a get function, meaning that only a call will be placed. Also, it is a public function so all parties would be able to see the result. No requirements, or modifiers are coded. The result of this function can only be visualized either as part of the bytecode or in the console.



```
https://rinkeby.etherscan.io/tx/0xe29c1ee6e74080cb43ff37ff8c455242bb9e01eb356dd496de06903a9cde07d5

[block:5032346 txIndex:11] from:0xbe3...25941 to:IDC.registerProducer(string,string,string,string,bool) 0x90c...2d4d0
value:0 wei data:0x86a...00000 logs:1 hash:0xe29...e07d5

status              0x1 Transaction mined and execution succeed
transaction hash    0xe29c1ee6e74080cb43ff37ff8c455242bb9e01eb356dd496de06903a9cde07d5
from                0xbe3b7fa5643c7a655aa5e30511b284a7e9925941
to                  IDC.registerProducer(string,string,string,string,bool) 0x90ca8afbaf8ffb70c523425378ca21721442d4
                    d0
gas                 179338 gas
transaction cost    179338 gas
```

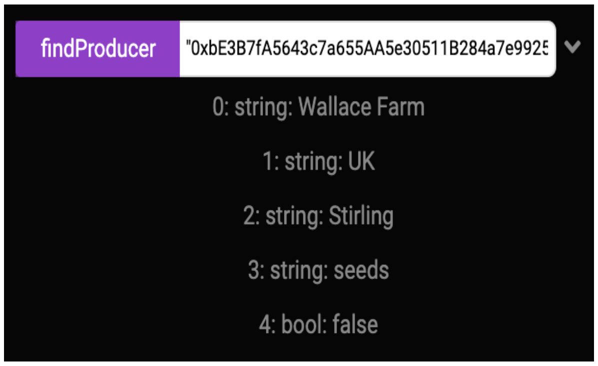


```
0xbE3B7fA5643c7a655AA5e30511B284a7e9925941,0x55345D
```

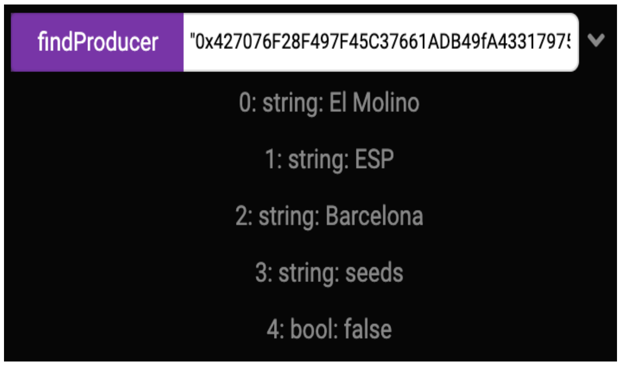
```
transaction hash

from
to
hash

input
decoded input
decoded output

logs
```

Find Producer / Crop / Shipment

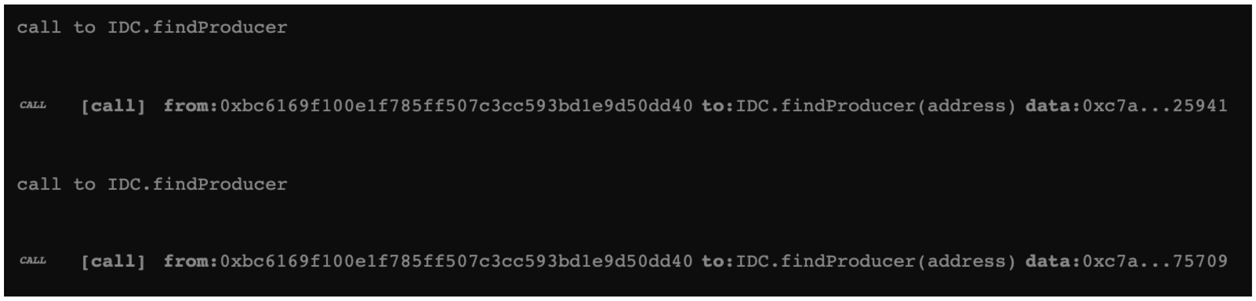
Again, a get function to find a producer, crop or shipment details through its address or ID respectively. Like the previous function, they are public with no requirements, or modifiers. The result of the functions can only be visualized either as part of the bytecode or in the Remix front.



```
findProducer  "0xbE3B7fA5643c7a655AA5e30511B284a7e9925
0: string: Wallace Farm
1: string: UK
2: string: Stirling
3: string: seeds
4: bool: false
```



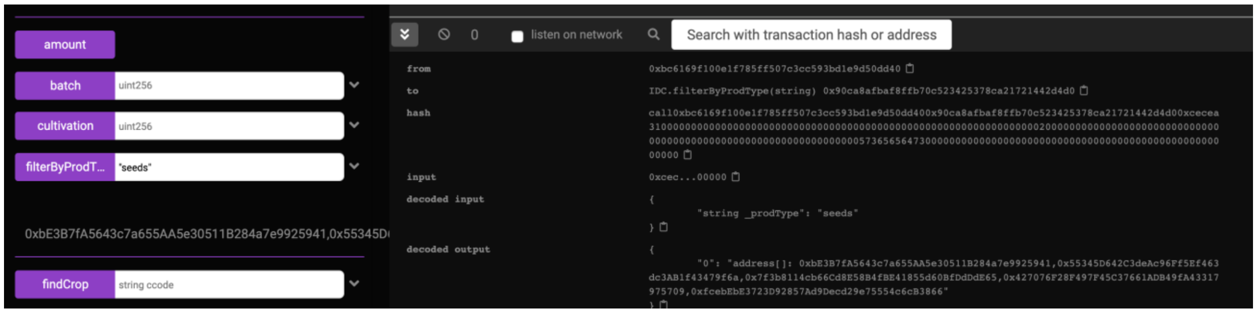
```
findProducer  "0x427076F28F497F45C37661ADB49fA4331797!
0: string: El Molino
1: string: ESP
2: string: Barcelona
3: string: seeds
4: bool: false
```



```
call to IDC.findProducer

CALL    [call]  from:0xbc6169f100e1f785ff507c3cc593bd1e9d50dd40 to:IDC.findProducer(address) data:0xc7a...25941

call to IDC.findProducer

CALL    [call]  from:0xbc6169f100e1f785ff507c3cc593bd1e9d50dd40 to:IDC.findProducer(address) data:0xc7a...75709
```

Filter by ProdType

Again, a get function to find a producer's detail based on the type of product associated to it. Like the previous function, it is a public with no requirements, or modifiers. The result of this function can only be visualized either as part of the bytecode or in the Remix front.

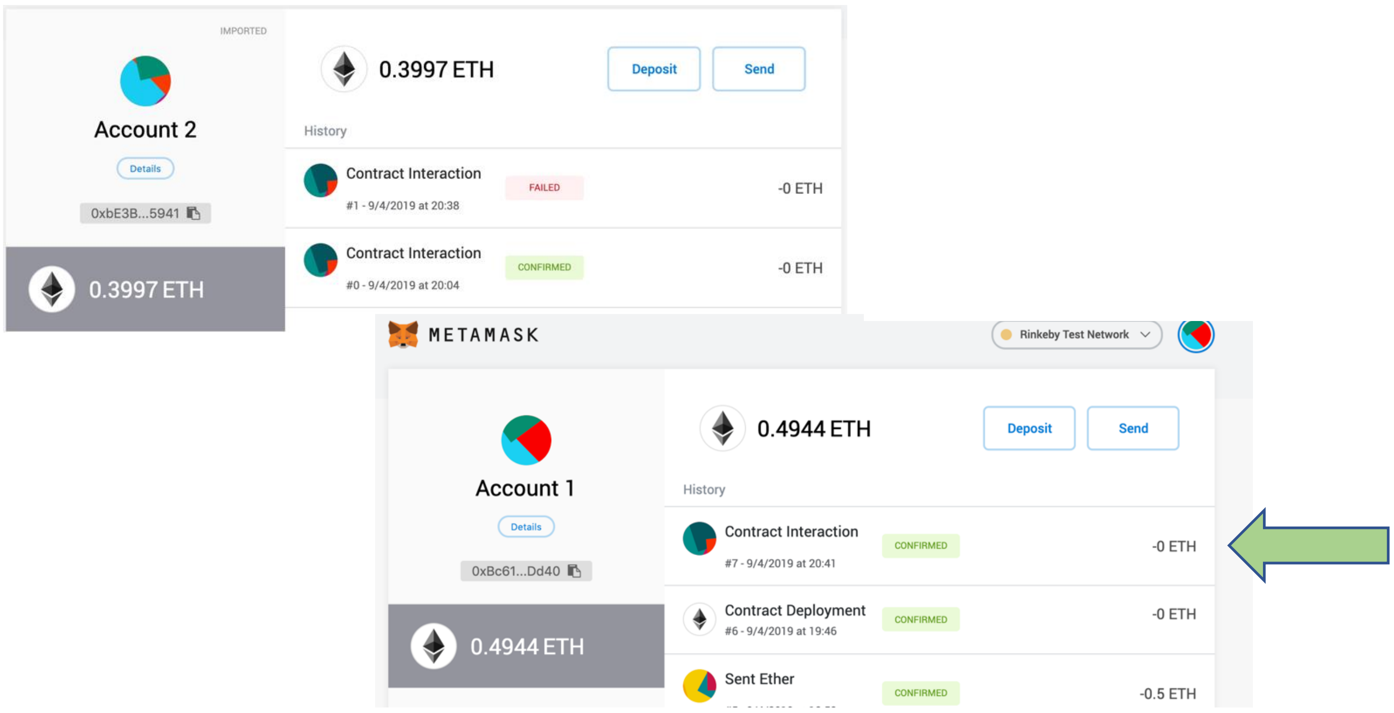


```
batch            uint256
cultivation      uint256
filterByProdT... "seeds"

0xbE3B7fA5643c7a655AA5e30511B284a7e9925941,0x55345D

findCrop         string ccode
```

```
from            0xbc6169f100e1f785ff507c3cc593bd1e9d50dd40
to              IDC.filterByProdType(string) 0x90ca8afbaf8ffb70c523425378ca21721442d4d0
hash            call0xbc6169f100e1f785ff507c3cc593bd1e9d50dd400x90ca8afbaf8ffb70c523425378ca21721442d4d00xcecea
                3100000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000
                0000000000000000000000000000000000573656564730000000000000000000000000000000000000000000000000
                00000
input           0xcec...00000
decoded input   {
                        "string _prodType": "seeds"
                }
decoded output  {
                        "0": "address[]: 0xbE3B7fA5643c7a655AA5e30511B284a7e9925941,0x55345D642C3deAc96Ff5Ef463
                dc3AB1f43479f6a,0x7f3b8114cb66Cd8E58B4fBE41855d60BfDdDdE65,0x427076F28F497F45C37661ADB49fA43317
                975709,0xfcebEbE3723D92857Ad9Decd29e75554c6cB3866"
                }
```

Remove Producer / Crop

These functions remove the registered status of a Producer or Crop albeit not "deleting" it from the database due to blockchain immutability property. A producer will no longer be able to register anything thereafter. These functions can only be called by the admin (Account 1), otherwise, the transaction will be reverted by the EVM.


IMPORTED

Account 2

Details

0xbE3B...5941

0.3997 ETH

Deposit Send

History

Contract Interaction FAILED -0 ETH
#1 - 9/4/2019 at 20:38

Contract Interaction CONFIRMED -0 ETH
#0 - 9/4/2019 at 20:04

METAMASK Rinkeby Test Network

Account 1

Details

0xBc61...Dd40

0.4944 ETH

Deposit Send

History

Contract Interaction CONFIRMED -0 ETH
#7 - 9/4/2019 at 20:41

Contract Deployment CONFIRMED -0 ETH
#6 - 9/4/2019 at 19:46

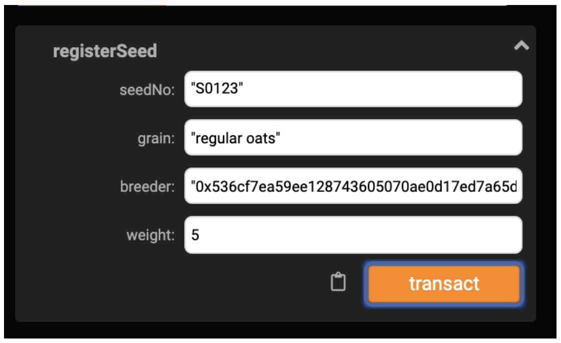
Sent Ether CONFIRMED -0.5 ETH

<u>Certify Producer</u>

This function confers a certified status to a producer. It can only be executed by the admin. However, additional requirements need to be established to obtain this level. Also, the function has work for improvements since the certified status is not updated as part of the producer's data.

<u>Register Seed / Crop/Harvest</u>

These functions share the same operating principle, likewise to register producer. Nonetheless, only a registered producer is able to interact with it, otherwise, the EVM will revert the transaction. Additional requirements such as avoiding duplicates and null values are also applied. For the crops, there is an additional validation regarding the amount of seeds (in kg) where the amount cropped cannot exceed the amount of seeds linked to it. Data is updated in the blockchain and an event is emitted.



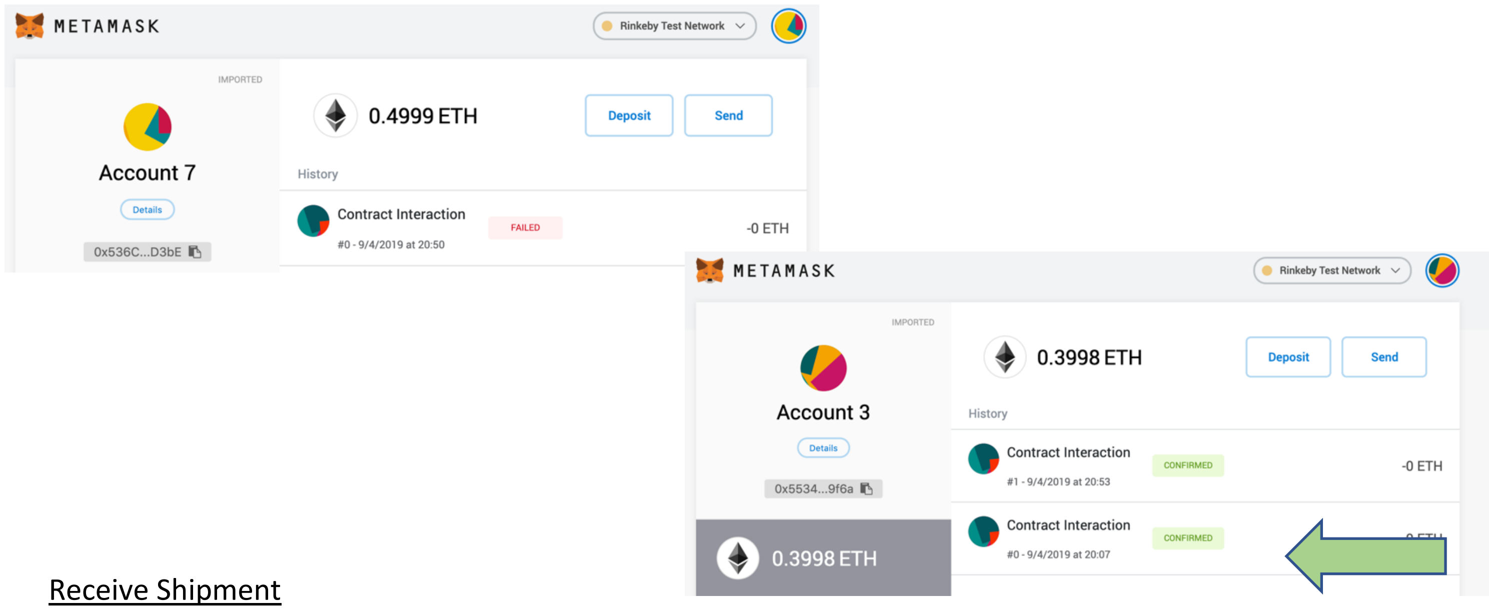
registerSeed

seedNo: "S0123"

grain: "regular oats"

breeder: "0x536cf7ea59ee128743605070ae0d17ed7a65d

weight: 5



Account 7

0x536C...D3bE

0.4999 ETH

Contract Interaction FAILED -0 ETH
#0 - 9/4/2019 at 20:50

Account 3

0x5534...9f6a

0.3998 ETH

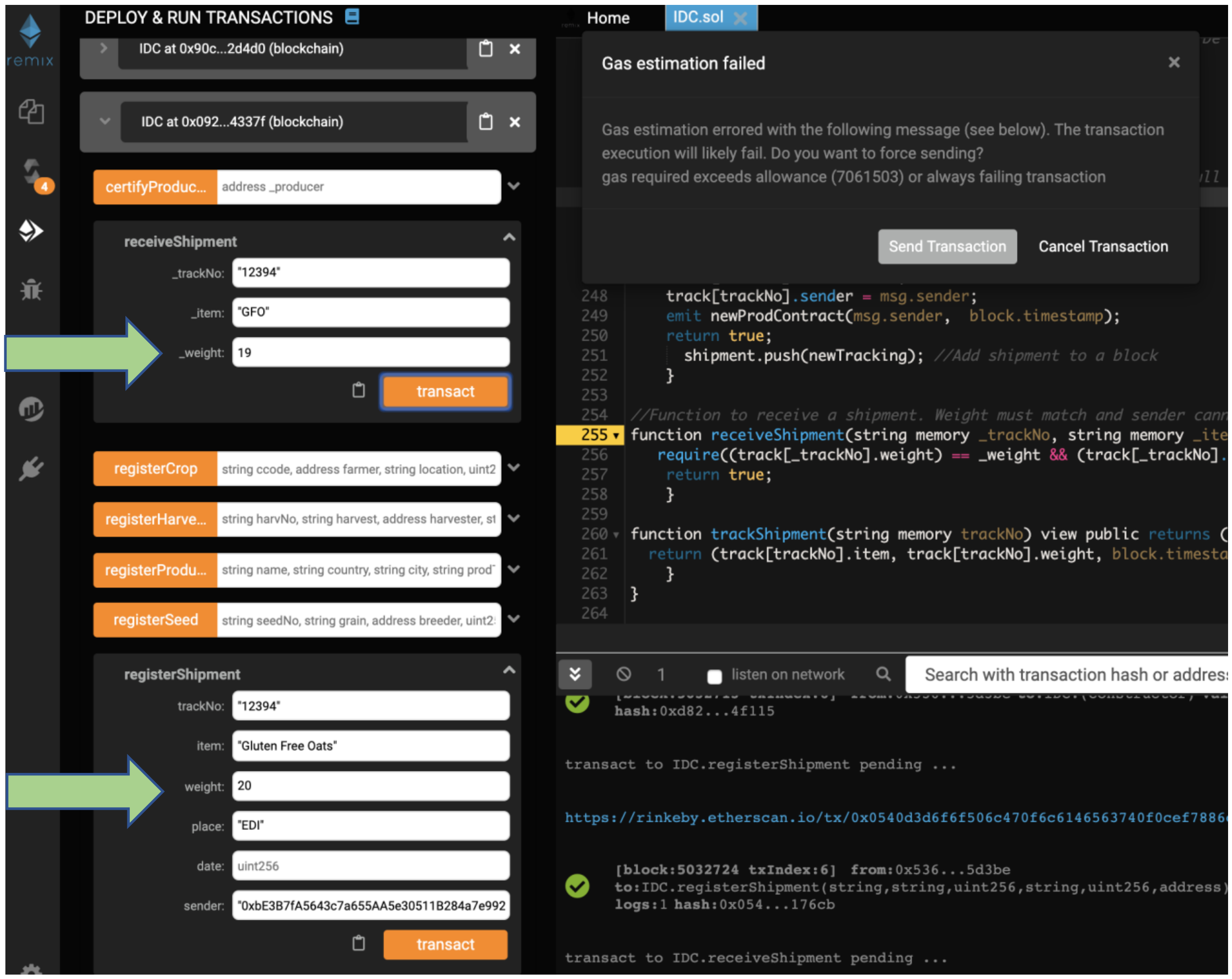
0.3998 ETH

Contract Interaction CONFIRMED -0 ETH
#1 - 9/4/2019 at 20:53

Contract Interaction CONFIRMED
#0 - 9/4/2019 at 20:07

Receive Shipment

Although the transportation and storage stages are not targeted in the current framework, this function intends to provide a source of trust for those products that are being shipped. The function requires the receiver address to be different from the sender; also, the weight from point A to point B should remain the same. The receiving transaction is stored on the blockchain and emitted as an event.


DEPLOY & RUN TRANSACTIONS

IDC at 0x90c...2d4d0 (blockchain)

IDC at 0x092...4337f (blockchain)

certifyProduc... address _producer

receiveShipment
_trackNo: "12394"
_item: "GFO"
_weight: 19
transact

registerCrop string ccode, address farmer, string location, uint2
registerHarve... string harvNo, string harvest, address harvester, st
registerProdu... string name, string country, string city, string prod
registerSeed string seedNo, string grain, address breeder, uint2

registerShipment
trackNo: "12394"
item: "Gluten Free Oats"
weight: 20
place: "EDI"
date: uint256
sender: "0xbE3B7fA5643c7a655AA5e30511B284a7e992
transact

Gas estimation failed

Gas estimation errored with the following message (see below). The transaction execution will likely fail. Do you want to force sending?
gas required exceeds allowance (7061503) or always failing transaction

Send Transaction Cancel Transaction

```
        track[trackNo].sender = msg.sender;
        emit newProdContract(msg.sender,  block.timestamp);
        return true;
          shipment.push(newTracking); //Add shipment to a block
        }

//Function to receive a shipment. Weight must match and sender cann
function receiveShipment(string memory _trackNo, string memory _ite
      require((track[_trackNo].weight) == _weight && (track[_trackNo].
        return true;
        }

function trackShipment(string memory trackNo) view public returns (
      return (track[trackNo].item, track[trackNo].weight, block.timesta
        }
}
```

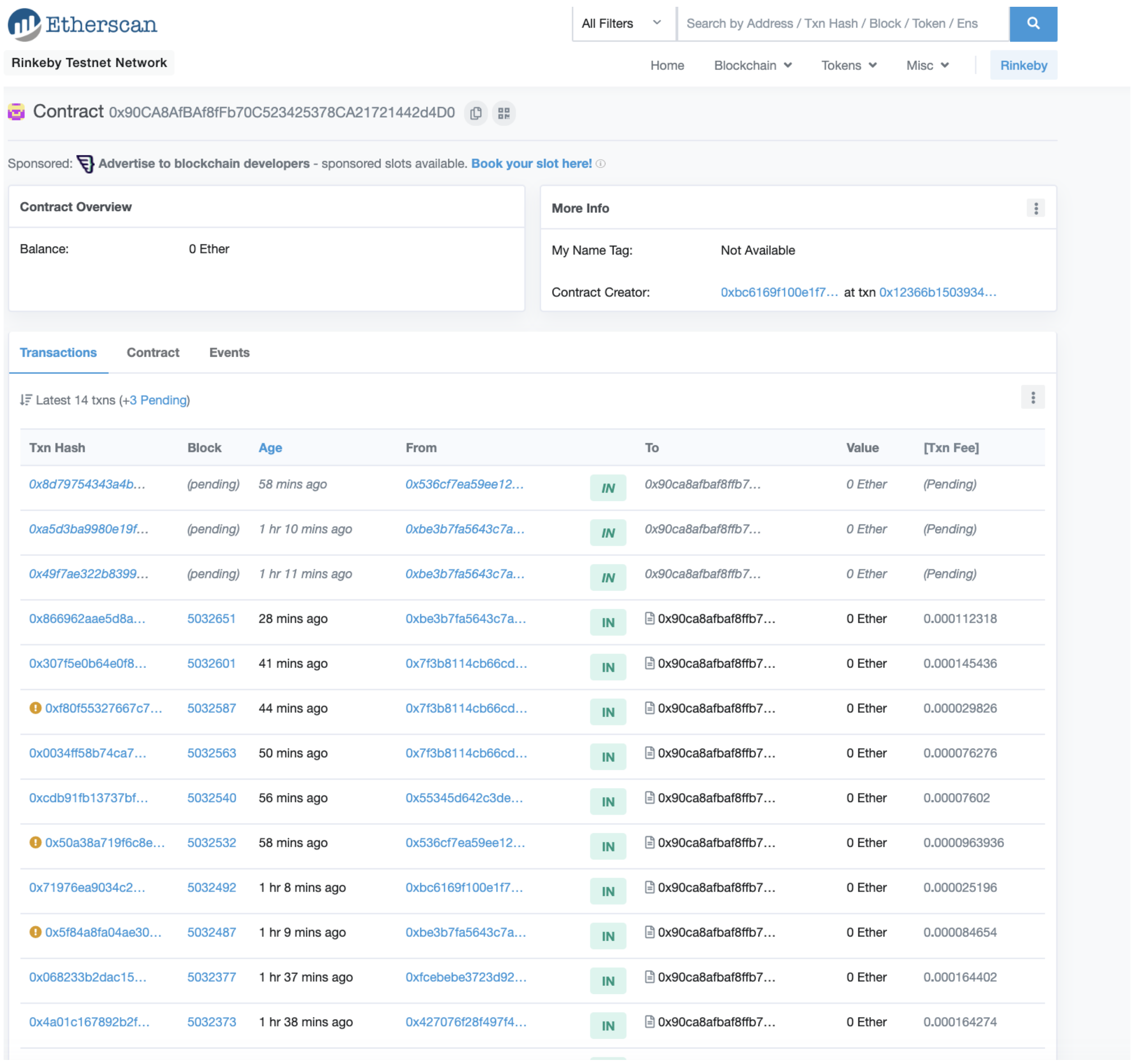
```
hash:0xd82...4f115

transact to IDC.registerShipment pending ...

https://rinkeby.etherscan.io/tx/0x0540d3d6f6f506c470f6c6146563740f0cef7886

    [block:5032724 txIndex:6]  from:0x536...5d3be
    to:IDC.registerShipment(string,string,uint256,string,uint256,address)
    logs:1 hash:0x054...176cb

transact to IDC.receiveShipment pending ...
```

The previous image throws an error due to inconsistency in both weights. Also, the same account that registered the shipment is invoking the function causing the EVM reversion.

5.3 Etherscan events

As revised in 3.2.8. Etherscan, will serve as a dashboard to visualize the transactions executed within the contract, as well as the events emitted by certain functions.

Once the contract is successfully deployed, a contract address will be generated. The contract information includes the transaction hash, the block where the transaction was included, the timestamp, the address of the sender, as well as the value and the corresponding fee.



Contract 0x90CA8AfBAf8fFb70C523425378CA21721442d4D0

Sponsored: **Advertise to blockchain developers** - sponsored slots available. **Book your slot here!**

Contract Overview

Balance: 0 Ether

More Info

My Name Tag: Not Available

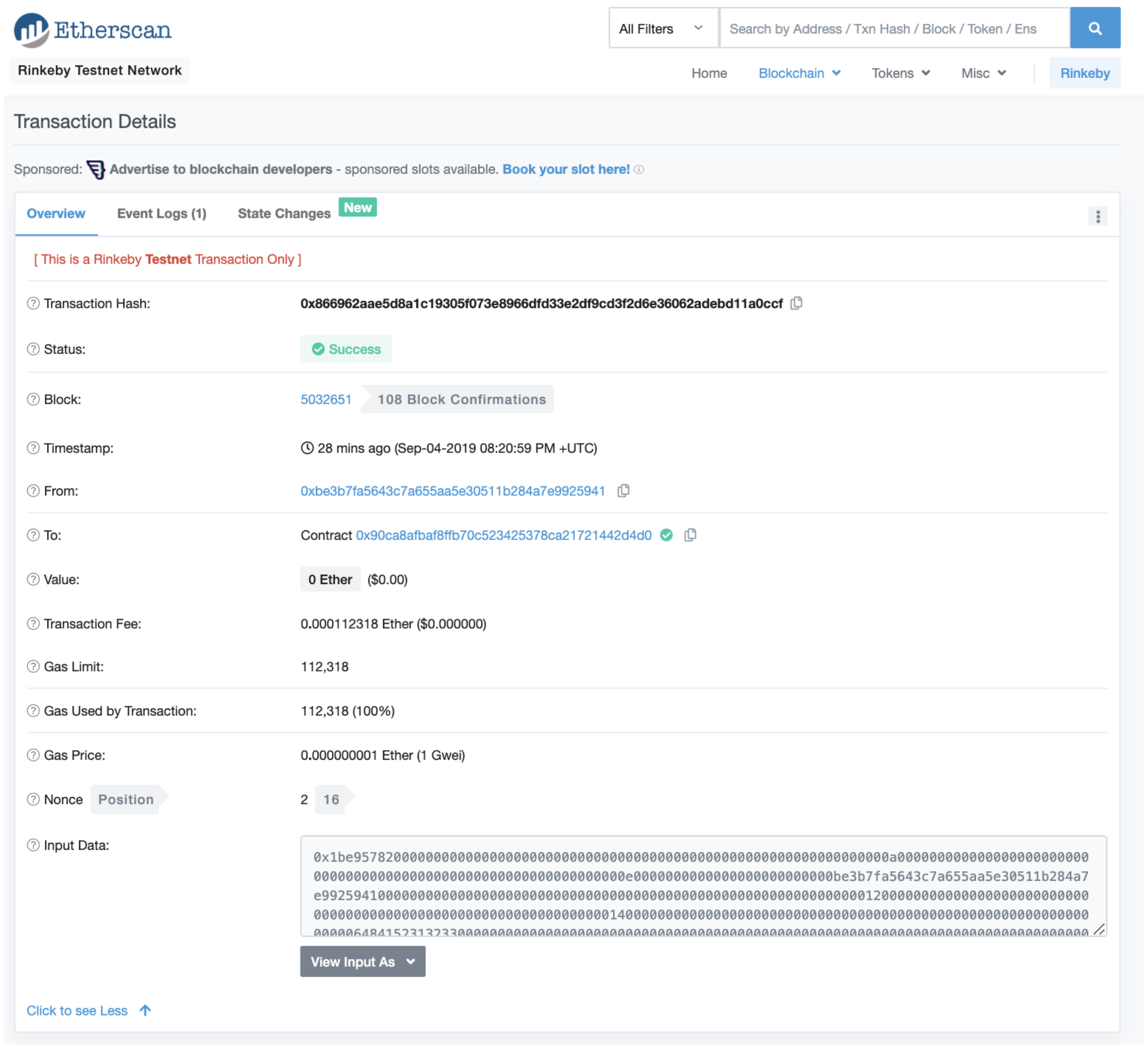
Contract Creator: 0xbc6169f100e1f7… at txn 0x12366b1503934…

Transactions Contract Events

Latest 14 txns (+3 Pending)

Txn Hash	Block	Age	From		To	Value	[Txn Fee]
0x8d79754343a4b…	*(pending)*	*58 mins ago*	*0x536cf7ea59ee12…*	*IN*	*0x90ca8afbaf8ffb7…*	*0 Ether*	*(Pending)*
0xa5d3ba9980e19f…	*(pending)*	*1 hr 10 mins ago*	*0xbe3b7fa5643c7a…*	*IN*	*0x90ca8afbaf8ffb7…*	*0 Ether*	*(Pending)*
0x49f7ae322b8399…	*(pending)*	*1 hr 11 mins ago*	*0xbe3b7fa5643c7a…*	*IN*	*0x90ca8afbaf8ffb7…*	*0 Ether*	*(Pending)*
0x866962aae5d8a…	5032651	28 mins ago	0xbe3b7fa5643c7a…	IN	0x90ca8afbaf8ffb7…	0 Ether	0.000112318
0x307f5e0b64e0f8…	5032601	41 mins ago	0x7f3b8114cb66cd…	IN	0x90ca8afbaf8ffb7…	0 Ether	0.000145436
! 0xf80f55327667c7…	5032587	44 mins ago	0x7f3b8114cb66cd…	IN	0x90ca8afbaf8ffb7…	0 Ether	0.000029826
0x0034ff58b74ca7…	5032563	50 mins ago	0x7f3b8114cb66cd…	IN	0x90ca8afbaf8ffb7…	0 Ether	0.000076276
0xcdb91fb13737bf…	5032540	56 mins ago	0x55345d642c3de…	IN	0x90ca8afbaf8ffb7…	0 Ether	0.00007602
! 0x50a38a719f6c8e…	5032532	58 mins ago	0x536cf7ea59ee12…	IN	0x90ca8afbaf8ffb7…	0 Ether	0.0000963936
0x71976ea9034c2…	5032492	1 hr 8 mins ago	0xbc6169f100e1f7…	IN	0x90ca8afbaf8ffb7…	0 Ether	0.000025196
! 0x5f84a8fa04ae30…	5032487	1 hr 9 mins ago	0xbe3b7fa5643c7a…	IN	0x90ca8afbaf8ffb7…	0 Ether	0.000084654
0x068233b2dac15…	5032377	1 hr 37 mins ago	0xfcebebe3723d92…	IN	0x90ca8afbaf8ffb7…	0 Ether	0.000164402
0x4a01c167892b2f…	5032373	1 hr 38 mins ago	0x427076f28f497f4…	IN	0x90ca8afbaf8ffb7…	0 Ether	0.000164274

Additional information is included at a transactional level, where details such as the gas price and limit, nonce, and bytecode among others can be found.



Transaction Details

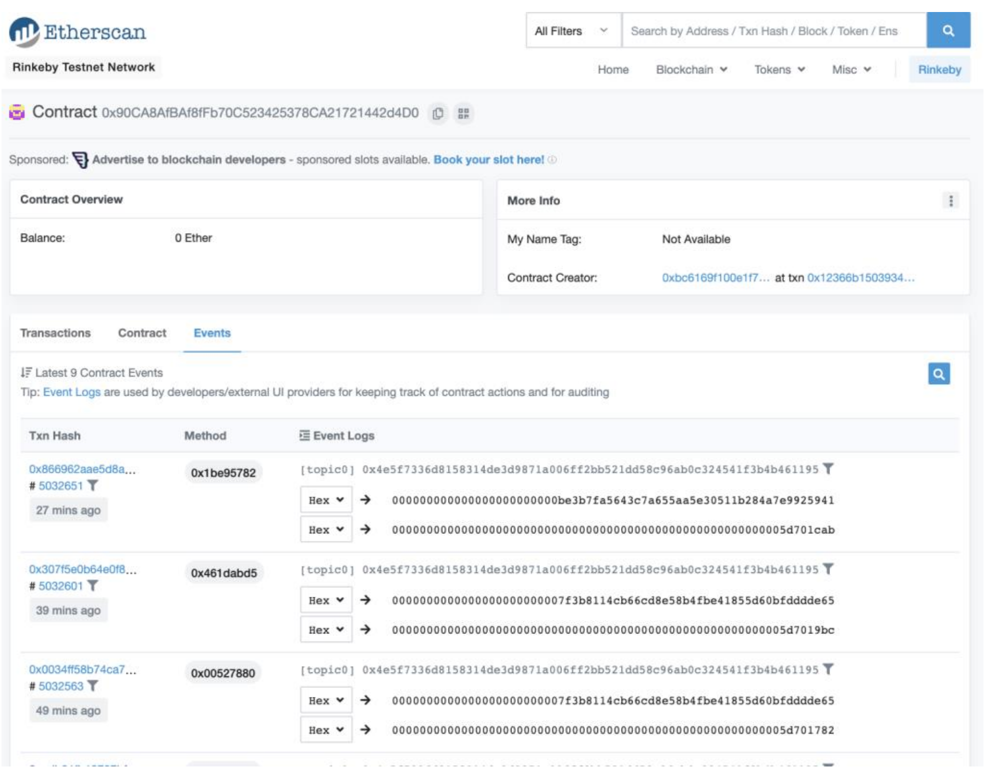
Sponsored: **Advertise to blockchain developers** - sponsored slots available. **Book your slot here!**

[This is a Rinkeby **Testnet** Transaction Only]

Transaction Hash:	0x866962aae5d8a1c19305f073e8966dfd33e2df9cd3f2d6e36062adebd11a0ccf
Status:	Success
Block:	5032651 108 Block Confirmations
Timestamp:	28 mins ago (Sep-04-2019 08:20:59 PM +UTC)
From:	0xbe3b7fa5643c7a655aa5e30511b284a7e9925941
To:	Contract 0x90ca8afbaf8ffb70c523425378ca21721442d4d0
Value:	0 Ether ($0.00)
Transaction Fee:	0.000112318 Ether ($0.000000)
Gas Limit:	112,318
Gas Used by Transaction:	112,318 (100%)
Gas Price:	0.000000001 Ether (1 Gwei)
Nonce Position	2 16
Input Data:	see below

```
0x1be9578200000000000000000000000000000000000000000000000000000000000000a000000000000000000000000
000000000000000000000000000000000000000e0000000000000000000000000be3b7fa5643c7a655aa5e30511b284a7
e9925941000000000000000000000000000000000000000000000000000000000000012000000000000000000000000000
000000000000000000000000000000000000140000000000000000000000000000000000000000000000000000000000
000006484152313233000000000000000000000000000000000000000000000000000000000000000000000000000000
```

Finally, information regarding the events emitted by a contract can also be accessed. This part is critical for the IDC project since these items constitute the bridge between the blockchain and the mobile application development. As of now, the IDC contract currently emits only test data (sender address and block timestamp in hexadecimal form) to verify that a connection can be established.


Etherscan
Rinkeby Testnet Network

All Filters | Search by Address / Txn Hash / Block / Token / Ens

Home | Blockchain | Tokens | Misc | Rinkeby

Contract 0x90CA8AfBAf8fFb70C523425378CA21721442d4D0

Sponsored: Advertise to blockchain developers - sponsored slots available. Book your slot here!

Contract Overview

Balance:	0 Ether

More Info

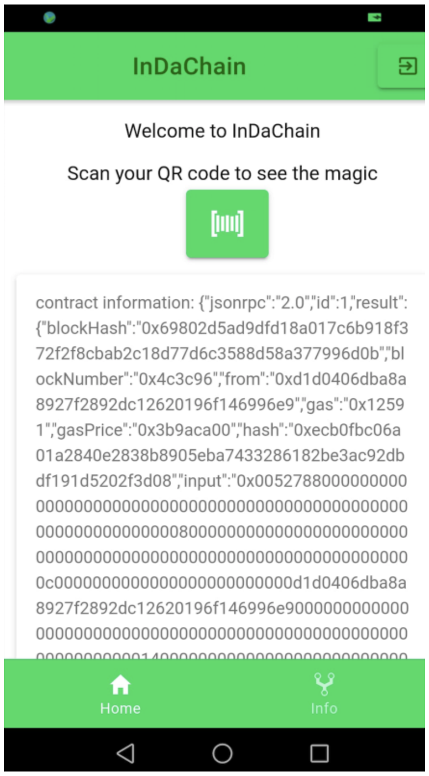
My Name Tag:	Not Available
Contract Creator:	0xbc6169f100e1f7... at txn 0x12366b1503934...

Transactions | Contract | Events

Latest 9 Contract Events
Tip: Event Logs are used by developers/external UI providers for keeping track of contract actions and for auditing

Txn Hash	Method	Event Logs
0x866962aae5d8a... # 5032651 27 mins ago	0x1be95782	[topic0] 0x4e5f7336d8158314de3d9871a006ff2bb521dd58c96ab0c324541f3b4b461195 Hex → 000000000000000000000000be3b7fa5643c7a655aa5e30511b284a7e9925941 Hex → 000000000000000000000000000000000000000000000000000000005d701cab
0x307f5e0b64e0f8... # 5032601 39 mins ago	0x461dabd5	[topic0] 0x4e5f7336d8158314de3d9871a006ff2bb521dd58c96ab0c324541f3b4b461195 Hex → 0000000000000000000000007f3b8114cb66cd8e58b4fbe41855d60bfdddde65 Hex → 000000000000000000000000000000000000000000000000000000005d7019bc
0x0034ff58b74ca7... # 5032563 49 mins ago	0x00527880	[topic0] 0x4e5f7336d8158314de3d9871a006ff2bb521dd58c96ab0c324541f3b4b461195 Hex → 0000000000000000000000007f3b8114cb66cd8e58b4fbe41855d60bfdddde65 Hex → 000000000000000000000000000000000000000000000000000000005d701782

Using the Etherscan integrated API [API], the mobile application is able to parse the data deployed by the contract, essentially functioning as the ABI.


InDaChain

Welcome to InDaChain

Scan your QR code to see the magic

```
contract information: {"jsonrpc":"2.0","id":1,"result":
{"blockHash":"0x69802d5ad9dfd18a017c6b918f3
72f2f8cbab2c18d77d6c3588d58a377996d0b","bl
ockNumber":"0x4c3c96","from":"0xd1d0406dba8a
8927f2892dc12620196f146996e9","gas":"0x1259
1","gasPrice":"0x3b9aca00","hash":"0xecb0fbc06a
01a2840e2838b8905eba7433286182be3ac92db
df191d5202f3d08","input":"0x005278800000000
0000000000000000000000000000000000000000000
0000000000000080000000000000000000000000
0000000000000000000000000000000000000000
0c000000000000000000000000d1d0406dba8a
8927f2892dc12620196f146996e900000000000
0000000000000000000000000000000000000000
```

6. Conclusion

Nowadays, the world relies more than ever on different production factors across several industries despite its questionable impact on the planet and society. All variety of goods and materials are being produced at alarming rates [59]. As a result, suppliers violate fundamental human rights, cause lasting environmental damage whilst exploiting the poor and powerless.

The inclusion of emerging technologies such as blockchain, that allows data to be trustworthy, interoperable and auditable, bring significant benefits and improvements into the operational nature of almost any business. Successful applications contribute to a culture shift providing tailored solutions for specific contexts.

Yet, there are important challenges ahead for adapting the traditional systems and processes into a more digital environment. Sustainable and scalable paradoxes surrounding the blockchain constrain the potential of the inclusion of blockchain and smart contracts in our daily habits. For example, as Aztori mentions, the current blockchain applications are generally not yet compatible with Internet of Things (IoT) networks since this type of devices possess a low computational capability when compared to a blockchain [60].

Still, most of times, “it’s the last key in the bunch that opens the door”. The IDC project is an idea that target the needs from two opposite ends across the supply chain process. First, it provides brands and retailers with a platform to show the world how their products and services are being delivered, either they are producing a natural or organic food or bringing a clean service that embraces sustainability. In the other end, it gets to consumers who are interested in understanding and can interpret information to learn where a product really comes from.

IDC intends to empower local business by showing the community true value in non-industrialized products, supporting the authenticity of valuable items, but most importantly, giving the customer an alternative to buy goods and services that halt harmful farming and production practices. All of these, achieved by bringing a true and transparent supply chain.

6.1 Evaluation

In general terms, the IDC smart contract is able to show how a production process could operate at a high-level when applied to a blockchain. As a whole, it creates a more transparent supply chain.

The deployment of the contract was achieved through Remix. Nevertheless, a robust testing should be applied before going into production in the main Ethereum network.

The contract is able to show some of the blockchain capabilities such as immutability (tamper-proof), decentralization and accessibility. The producers can record their own data, as well as their own goods information in the blockchain. The network participants can retrieve the registered attributes to verify whilst the contract is accessible to the participants in the network. Also, a producer can achieve a certified status although the logic.

Moreover, in terms of tracking, the contract allows to record the shipment of a product. In the blockchain Upon arrival, the receiving party can confirm whether the product arrived in the expect conditions. The contract also allows individuals to retrieve a shipment information.

On the other hand, the contract still has room for improvement.

- First, the contract functions can be optimized. A smart contract objective should target on storing and keeping a trustful source of information and implementing a simple process rather than performing complex math since most of operations occurring inside a contract incur in a cost.
- The Certification logic currently allows the contract administrator to gran the certified status to a certain producer. However, the data of the producer is not being updated in the blockchain yet. Additional requirements can also be included as part of the certification rationale.
- Once the producer establishes the product that is being recorded, the contract should automatically invoke another contract based on the type of good to complete the registration.

- Currently, the events emitted only show information related to the block where a transaction is being stored per se. The inclusion of data more useful for users and producers should be integrated.

6.2 Limitations

- The most important limitation experienced was connecting the blockchain to the mobile application front-end scripts using Web3. At the time of deploying a contract via Web3 with Infura, the resulting ABI from the operation could not be parsed into the JavaScript front-end files.
- Due to consistent updates in the technologies used, several inconsistencies were experienced during the development of the project causing setbacks on the original plan. Remix and Web3 were the most affected parties.
- When deployed through Web3, the provider could not be properly set up at times, prompting a change in the way of deployment to Remix. In the end, Remix was a very useful tool due to the versatility at the moment of demonstrating the results in a friendlier manner.
- Although a real production process was studied and applied to the contract development, the data and logic at some points was completely simulated.
- The contract was not tested using an installed Ethereum Client such as Geth or Parity.

6.3 Framework for the Future

If certain factors are aligned, the potential of IDC or any other related initiative is significant. By including smart censors and Enterprise Resource Planning systems, Provenance and Coop are working together in an initiative that empowers customers with knowledge of the true origin of the goods they are acquiring [61]. IBM and Maersk [62] are joining efforts to reduce international barriers in logistics by providing an end to end traceable shipping platform.

Based on the previous work, IDC can evolve into a digital integrated environment by adding three layers of solutions to a traditional supply chain process (figure 23). The future framework considers first, the inclusion of technologies such as IoT devices, smart censors and Near-Field

Communication devices. By adding this layer, traceability is enhanced along a product's journey while special requirements like temperatures and geo-location data can be enabled.

A second layer considers the inclusion of mathematical algorithms and infrastructure techniques to optimize and create a more efficient data life cycle. Also, by developing machine learning algorithms, several processes can be optimized resulting in a more efficient way of production.

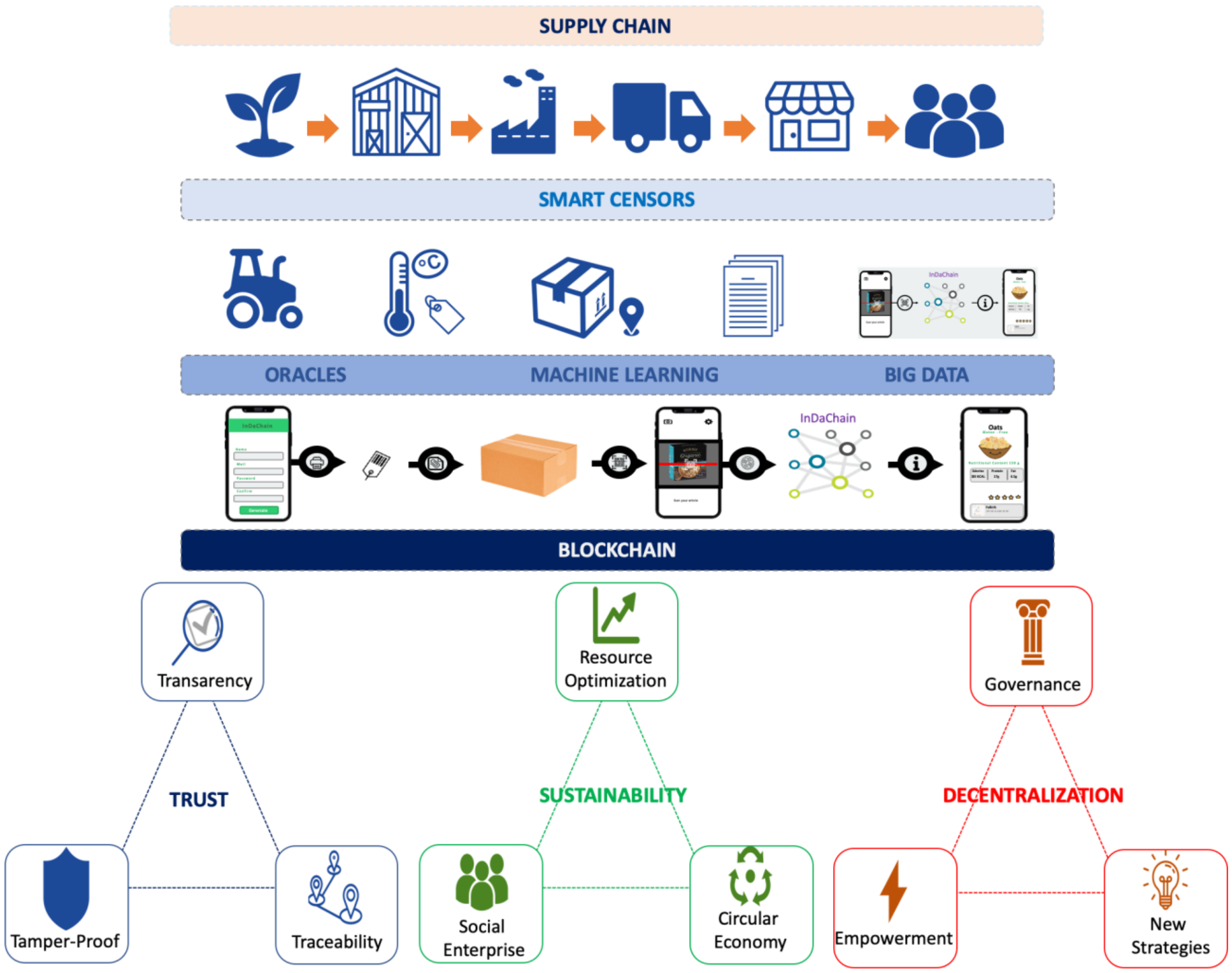


Figure 23: IDC future framework for Supply Chain

Finally, the third layer is the blockchain. Smart contracts are key for a transparent and trustful supply chain. Moreover, by connecting to the previous layers, new capabilities can be achieved. One example can be oracles, which connected to smart contracts by Python scripts, can create a real-time intelligent ecosystem based on three main foundations: becoming a trustworthy source of information; create a decentralized system; and achieve a sustainable way of production.

References

[1] McGovern, M. (2017, April 13). Creating a thinking supply chain for the Cognitive Era. *Watson Customer Engagement*. Retrieved from https://www.ibm.com/blogs/watson-customer-engagement/2017/03/27/thinking-supply-chain/

[2] Hutchins, M. J., & Sutherland, J. W. (2008). An exploration of measures of social sustainability and their application to supply chain decisions. *Journal of Cleaner Production, 16*(15), 1688-1698. https://doi.org/10.1016/j.jclepro.2008.06.001

[3] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Retrieved from https://nakamotoinstitute.org/bitcoin

[4] Botjes, E. (2017, August 11). Pulling the Blockchain apart. The transaction life-cycle. *Unlock innovation*. Retrieved from https://medium.com/ignation/pulling-the-blockchain-apart-the-transaction-life-cycle-7a1465d75fa3

[5] Temitayo, A. (2019, June 30). EDC Blockchain. *Medium*. Retrieved from https://medium.com/@temitayoadebanjo2/every-day-coin-edc-blockchain-is-a-blockchain-technology-that-uses-the-more-advanced-form-of-76e354bc17f7

[6] Crosby, M., Nachiappan., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain technology: Beyond Bitcoin. *Applied Innovation Review*. Retrieved fromhttps://j2-capital.com/wp-content/uploads/2017/11/AIR-2016-Blockchain.pdf

[7] Housley, R. (2004). Public Key Infrastructure (PKI). *John Wiley & Sons, Inc*. https://doi.org/10.1002/047148296X.tie149

[8] Dwork, C., & Naor, M. (2001). Pricing via Processing or Combating Junk Mail. *Annual International Cryptology Conference*, 139-147. https://doi.org/10.1007/3-540-48071-4_10

[9] Digiconomist. Bitcoin Energy Consumption Index [Blog]. Retrieved from https://digiconomist.net/bitcoin-energy-consumption

[10] Lock, C., & Wright, A. (2011, January 18). Typical domestic energy consumption figures). *Office of Gas and Electricity Markets*. Retrieved from https://www.ofgem.gov.uk/ofgem-publications/64026/domestic-energy-consump-fig-fs-pdf

[11] Barnard, M. (2018, December 8). The dark side of blockchain: Electricity Consumption (blockchain report excerpt). *CleanTechnica*. Retrieved from https://cleantechnica.com/2018/12/08/the-dark-side-of-blockchain-electricity-consumption-blockchain-report-excerpt/

[12] Daily Hodl (2018, April 27). Cryptocurrency Transaction Speeds: The Complete Review. *The daily Hodl: News and insigh for the Digital Economy*. Retrieved from https://dailyhodl.com/2018/04/27/cryptocurrency-transaction-speeds-the-complete-review/

[13] Yli-Huumo, J., Ko, D., Choi, S., Park, S., & Smolander, K. (2016). Where is current research on Blockchain Technology? – A Systematic Review. *PLoS ONE, 11*(10):e0163477. https://doi.org/10.1371/journal.pone.0163477

[14] Madisetti, V., & Bahga, A. (2018). Method and system for tuning blockchain scalability for fast and low-cost payment and transaction processing. Retrieved from https://patents.google.com/patent/US10102265B1/en

[15] Rosic, A. (2016). Smart contracts: The blockchain technology that will replace lawyers. BlockGeeks. Retrievedf rom https://blockgeeks.com/guides/smart-contracts/.

[16] Tokia. Proof of work and proof of Stake [Blog]. Retrieved from https://www.tokia.io/blog/learn/what-is-pow-pos/

[17] Saleh, F. (2019). Blockchain without waste: Proof-of-stake. http://dx.doi.org/10.2139/ssrn.3183935

[18] Buterin, V. (2013). A next generation smart contract & decentralized application platform. *Ethereum White Paper*. Retrieved from https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf

[19] DistrictOx Educational Portal. (2019). What Is Ethereum?. Retrieved from **Error! Hyperlink reference not valid.**https://education.district0x.io/general-topics/understanding-ethereum/what-is-ethereum/

[20] Li, X., Jiang, P., Chen, T., Luo, X., & Wen, Q. (2017). A survey on the security of blockchain systems. *Future Generation Computer Systems.* https://doi.org/10.1016/j.future.2017.08.020

[21] Swan, M. (2015). Blockchain: Blueprint for a New Economy. *O'Reilly Media, Inc*. Retrieved from https://www.oreilly.com/library/view/blockchain/9781491920480/ch01.html

[22] Etherscan. (n.d.). Ethereum Block Time History-Ethereum Avarage Block Time chart. *Etherscan*. Retrieved from https://etherscan.io/chart/blocktime

[23] Buterin, V., & Griffith, V. (2017). Casper the Friendly Finality Gadget. Retrieved from https://arxiv.org/pdf/1710.09437.pdf

[24] Ethereum Research. (2017). Future-compatibility for sharding [Blog]. Retrieved from https://ethresear.ch/t/future-compatibility-for-sharding/386

[25] Tsao, P. (2018, July 31). Blockchain 2.0 and Ethereum [Blockchain Basics Part 3]. *Medium*. Retrieved from https://medium.com/xpa-2-0/blockchain-2-0-and-ethereum-blockchain-basics-part-3-362eb3561b4e

[26] Delmolino, K., Arnett, M., Kosba, A., Miller, A., & Shi, E. (2016). Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab. *International Conference on Financial Cryptography and Data Security.* https://doi.org/10.1007/978-3-662-53357-4_6

[27] Grech, N., Kong, M., Jurisevic, A., Brent, L., Scholz, B., & Smaragdakis, Y. (2018). MadMax: Surviving Out-of-Gas Conditions in Ethereum Smart Contracts. *Proc. ACM Program. Lang. 2, OOPSLA*, *2*(16), 116-143. https://doi.org/10.1145/3276486

[28] Iyer, A., & Dannen, C. (2018). Conceptual Introduction. *Building Games with Ethereum Smart contracts*, 1-17. https://doi.org/10.1007/978-1-4842-3492-1_1

[29] Ethos. What is Ethereum Gas?. Retrieved from https://www.ethos.io/what-is-ethereum-gas/

[30] Boily, F. (2018, May 14). Explaining Ethereum test networks and all their differences. *Medium.* Retrieved from https://medium.com/coinmonks/ethereum-test-networks-69a5463789be

[31] Dinçer, H., & Yüksel, S. (2019). Handbook of Research on Managerial Thinking in Global Business Economics. Hershey, PA: IGI Global. DOI:10.4018/978-1-5225-7180-3

[32] Siegel, D. (2016). Understanding the DAO Attack. *Coindesk.* Retrieved from https://www.coindesk.com/understanding-dao-hack-journalists

[33] Akentiev, A. (2017, November 8). Parity Multisig Hacked. Again. *Medium*. Retrieved from https://medium.com/chain-cloud-company-blog/parity-multisig-hack-again-b46771eaa838

[34] Jules, A., Kosba, A., & Shi, E. (2016). The ring of Gyges: Using smart contracts for crime. In: SIGSAC conference on computer and communications security, 283–295. Retrieved from http://www.arijuels.com/wp-content/uploads/2013/09/Gyges.pdf

[35] Atzei, N., Bartoletti, M., & Cimoli, T. (2017). A survey of attacks on Ethereum Smart Contracts (SoK). *Principles of Security and* Trust, 164-186. DOI: 10.1007/978-3-662-54455-6 8

[36] Zyskind, G., Nathan, O., & Pentland, A. (2015). Decentralizing Privacy: Using Blockchain to Protect Personal Data. *IEEE Security and Privacy Workshops*, 180-184. DOI: 10.1109/SPW.2015.27

[37] Paul, G., Sarkar, P., & Mukherjee. S. (2014). Towards a More Democratin Mining in Bitcoins. *International Conference on Information Systems Security*, 185-203. https://doi.org/10.1007/978-3-319-13841-1_11

[38] Dagher, G. G., Marella, P. B., Milojkovic, M., & Mohler, J. (2018). BroncoVote: Secure Voting System using Ethereum's Blockchain. *International Conference on Information Systems Security and Privacy*, *(4),* 96-107. DOI: 10.5220/0006609700960107

[39] Artlyst. (2015, January 20). Spanish forgery ring creating Picasso, Miró, and Matisse fakes arrested. Retrieved from https://www.artlyst.com/news/spanish-forgery-ring-creating-picasso-miro-and-matisse-fakes-arrested/

[40] Lawrence, F. (2013 February 15). Horsemeat scandal: the essential guide. *The guardian*. Retrieved from https://www.theguardian.com/uk/2013/feb/15/horsemeat-scandal-the-essential-guide

[41] Roberts, J.J. (2017, September 12). The Diamond Industry Is Obsessed with the Blockchain. Fortune. Retrieved from http://fortune.com/2017/09/12/diamond-blockchain-everledger/

[42] Baker, J. (2017, May 15). The story of Provenance: The blockchain startup revolutionising supply chains. *Project Breakthrough.* Retrieved from http://breakthrough.unglobalcompact.org/briefs/jessi-baker-provenance-the-blockchain-startup-revolution/

[43] Project Provenance Ltd. (2015, November 21). Blockchain: the solution for transparency in product supply chains. *Provenance*. Retrieved from https://www.provenance.org/whitepaper

[44] Toyoda, K., Mathiopoulos, P. T., Sasase, I., & Ohtsuki, T. (2017). A Novel Blockchain-Based Product Ownership Management System (POMS) for Anti-Counterfeits in the Post Supply Chain. IEEE Access. DOI: 10.1109/ACCESS.2017.2720760

[45] Iyer, K., & Dannen, C. (2018). Crypto-economics and Game Theory. *Building Games with Ethereum Smart Contracts, 129-141.* https://doi.org/10.1007/978-1-4842-3492-1_6

[46] Digital Ocean (n.d.). Remix-IDE Layout. *Ethereum Revision*. Retrieved from https://remix-ide.readthedocs.io/en/latest/layout.html

[47] Hirai, Y. (2017). Defining the Ethereum Virtual Machine for Interactive Theorem Provers. *International Conference on Financial Cryptography and Data Security,* 520-535. DOI: 10.1007/978-3-319-70278-0_33

[48] Digital Ocean (n.d.). Introduction to Smart Contracts. *Ethereum Revision*. Retrieved from https://solidity.readthedocs.io/en/v0.5.11/introduction-to-smart-contracts.html#index-6

[49] Zheng, Z., Xie, S., Dai, H., Chen, X., & Want, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trend. *IEEE International Congress on Big Data (BigData Congress), 557-564. DOI*: 10.1109/BigDataCongress.2017.85

[50] Digital Ocean (n.d.). Solidity. *Ethereum Revision*. Retrieved from https://solidity.readthedocs.io/en/v0.5.11/

[51] MetaMask. (2019, August 21). Getting Started. *MetaMask*. Retrieved from https://metamask.github.io/metamask-docs/Main_Concepts/Getting_Started

[52] Curran, B. (2019, February 6). What is Ethereum's Infura? Scalable Access to Ethereum and IPFS. *Blockonomi*. Retrieved from https://blockonomi.com/ethereum-infura/

[53] Web3. Getting Started. *Ethereum Revision*. Retrieved from https://web3js.readthedocs.io/en/v1.2.1/getting-started.html

[54] Truffle Suite (n.d.). Drizzle-Fresh chaing-data for front-ends. *Truffle Blockchaing Group*. Retrieved from https://www.trufflesuite.com/drizzle

[55] Truffle Suite (n.d.). Ganache-One Click Blockchain. *Truffle Blockchaing Group*. Retrieved from https://www.trufflesuite.com/ganache

[56] Truffle Suite (n.d.). Truffle-Smart contracts made sweeter. *Truffle Blockchaing Group*. Retrieved from https://www.trufflesuite.com/truffle

[57] Zhu, N. (2018, August 25). 5 minute guide to deploying smart contracts with Trufle and Ropsten. *Medium*. Retrieved from https://medium.com/coinmonks/5-minute-guide-to-deploying-smart-contracts-with-truffle-and-ropsten-b3e30d5ee1e

[58] Wood, G. (2014). Ethereum: A secure decentralized generalized transaction ledger. Retrieved from http://gavwood.com/paper.pdf

[59] Earth Overshoot Day (nd.d). Country Overshoot Days*. Global Footprint Network.* Retrieved from https://www.overshootday.org/newsroom/country-overshoot-days/

[60] Atzori, L., Iera, A., & Morabito, G. (2016). Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Networks.* DOI: 10.1016/j.adhoc.2016.12.004

[61] Provenance. (n.d). Pioneering a new standard for trust in food retail. Project Provenance Ltd. Retrieved from https://www.provenance.org/case-studies/co-op

[62] White, M. (2018, January 16). Digitizing Global Trade with Maersk and IBM. Blockchain Pulse: IBM Blockchain Blog. Retrieved from https://www.ibm.com/blogs/blockchain/2018/01/digitizing-global-trade-maersk-ibm/

Appendix: Contract Boilerplate

Variables		
Name	**Type**	**Description**
admin	address	Address of the person managing the cultivation process
producers	mapping	List of addresses of registered producers
seeds	mapping	List of registered seeds mapped by ID
crops	mapping	List of registered crops mapped by ID
harv	mapping	List of registered harvests mapped by ID
track	mapping	List of registered products mapped by ID
shipments	mapping	List of shipments weight corresponding to a product
senders	mapping	List of addresses confirmed as registered producers
regProds	mapping	Match a value address from a boolean key
seed	Seed []	An array of a Seed struct used to add entered values into a block
supplier	Producer []	An array of a Producer struct used to add entered values into a block
cultivation	Crop []	An array of a Crop struct used to add entered values into a block
batch	Harvest []	An array of a Harvest struct used to add entered values into a block
shipment	Tracking []	An array of a Tracking struct used to add entered values into a block
amount	uint	Undefined integer used to monitor the weights of seeds, products and shipments
producerAddress	address []	Array of addresses that can be stored as memory values to retrieve a list of registered producers

Seeds Struct		
Name	**Type**	**Description**
grain	uint	Seed ID
farmer	address	Who is registering a seed
Amount	uint	Amount of seeds in kg.

Producer Struct		
Name	**Type**	**Description**
name	string	Producer name
country	string	Country of origin
city	string	City of origin
prodType	string	Type of product
certified	boolean	Certified Producer?

Crop Struct		
Name	**Type**	**Description**
farmer	address	Who is registering a crop
location	string	Where is the crop cultivated
quantity	uint	Amount of cropped seeds in kg
sowings	boolean	Certified Seed?
prevcrops	uint	Number of previous croppings
fertilizer	string	Agro-chemicals used
timestamp	uint	Block data

Harvest Struct		
Name	**Type**	**Description**
harvest	string	Harvest ID
harvester	address	Who is registering the harvest?
field	string	Place of harvest
harvquant	uint	Amount of oats harvested in kg

Modifier	
Name	**Description**
restricted	Modifier that enables the right to execute a certain function only to the admin
registered	Modifier that requires an address to be already stored

Functions	
Name	**Description**
Constructor	Constructor function to deploy an instance of the contract
registerProducer	Called to register as a Producer
allSuppliers	Called to retrieve a list of registered producers
findProducer	Function to find a producer by its address
filterByProdType	Called to retrieve a list of products by type
removeProducer	Called to remove a registered producer from the database
certifyProducer	Called to certify a producer if required
registerSeed	Called to register a Seed
findSeed	Function to find a seed by its ID
registerCrop	Called to register a Crop
findCrop	Function to find a Crop by its ID
removeCrop	Called to remove a crop from database
registerHarvest	Called to register a Harvest batch
registerShipment	Called to register the shipment of a product
receiveShipment	After a shipment is received, called to verify product and shipment data
trackShipment	Function to find a shipment by ID