Massachusetts Institute of Technology
Media Lab's Digital Currency Initiative
Sloan School of Management

Oracle for Smart Cities With Use Case: Parametric Flight Delay Insurance

Authors: James Fok - SFMBA 2019, Mark Adams - MBA 2019, Santhosh, Narayan - MIT EECS MENG 2019, and Thomaz do Nascimento - SFMBA 2019

Keywords: Oracles, Smart cities, Parametric Flight Delay Insurance

Blockchain Lab Program
2018-2019 Working Groups Cycle
Taught by N. Narula, G. Gensler, S. Johnson, and M. Casey
Member Company: Monetary Authority of Singapore
Project Group: Trusted Oracles for Smart Cities

DCI Working Group

Oracle for Smart Cities

With Use Case:
Parametric Flight Delay Insurance

15.S68 - Blockchain Lab

James Fok - SFMBA 2019

Mark Adams - MBA 2019

Santhosh Narayan - MIT EECS MENG 2019

Thomaz do Nascimento - SFMBA 2019

In collaboration with:

The Monetary Authority of Singapore

digital currency initiative

Executive Summary

Oracles and smart contracts can create and deliver value for parametric insurance by eliminating current transactional frictions and opening up new opportunities. The challenges are both technical and behavioral. The key to success is in navigating these challenges methodically and making the right decisions in system design and specification.

In basic contract law, a judicial system adjudicates contractual disputes and enforces terms of the agreement. Meanwhile, smart contracts are enforced by built-in code or in other words, “code is law.” Smart contracts have the potential to speed up contract execution (i.e. transactions) and empower individuals to become independent from bureaucracies while maintaining the reliability of these institutions.

One potential use-case of these smart contracts is in parametric insurance. Parametric insurance, unlike traditional insurance, revolves around a pre-agreed payout based on projected loss and probability that is automatically triggered when a parameter is met (e.g. flight delay) based on inputs from a trusted source. It has logical applications in the flight delay use case because it can easily be purchased digitally, separately or bundled with tickets, and there are limited hassles to claim the benefits since it’s paid out to travelers as soon as the delay is confirmed. Thanks to logical alignment, it also stands to benefit greatly from the use of smart contracts, which is why this was selected as our focus use case in collaboration with MAS.

A major barrier to ubiquitous smart-contract use is determining how code can acquire information to process from the real-world; this is known as the Oracle problem. The Oracle problem can be divided in two parts: 1) arbitration and 2) submission. Arbitration governs the logic which the Oracle uses to make an objective decision. Submission is the technological challenge that Oracle faces in securely and reliably obtaining the information. Through the exploration of these challenges in this project, solutions were found to address each of these issues.

In the proposed framework to resolve the arbitration problem, the Oracle design follows two particular dimensions, incentive alignment and trustworthiness. Information can be broadly

categorized into three non-exclusive sources: 1) centralized governments, 2) trusted entities, and 3) permissionless crowdsourced data, with decreasing level of intrinsic trust.

Incentives must be aligned for all parties involved in order for the algorithmic approach to arbitration to work. In the absence of aligned incentives, parties would almost certainly deviate from reporting the “true” state of affairs in order to obtain financial gains. The ways in which alignment can be achieved are through system design and the use of technology. That is to create competing incentives to counter misalignment and to deploy technologies, such as IoT or augmented geolocation information that can bypass or calibrate against bad behaviours.

From the trust perspective, the arbitration mechanism can be designed to work with varying levels of intrinsic trust that characterises the three types of information sources. It is important to note that whilst the algorithm is objective, the perceived trustworthiness is subjective. Government information would normally be considered trustworthy but those wishing to disintermediate centralised control would prefer another source. From the insurers’ standpoint this subjectivity can be a source of opportunity, utilising smart contracts to create parallel contracts and products that tailor for individual needs without incurring substantial investment.

Another issue relating to this mechanism is when decisions are made with multiple sources involved. Trust continues to play an important role but even if there is an identical level of trust, a decision algorithm is still necessary to arbitrate between competing information. Starting with the most basic form using simple majority to getting an unanimous decision, all of these remain subjective to the individuals. Insurers should design the mechanism with market dynamics in mind.

Commercial Considerations

The use of blockchain and smart contracts in parametric flight insurance is very limited at a commercial level at this point in time, apart from a few pilots from large insurance companies and insurtech startups. Hurdles to commercialization include the availability of quality input data and Oracles, as well as the cost benefit trade-off: it’s unlikely the value to customers of additional transparency and trust are worth the added cost of running these systems on the blockchain rather than in a database when customers are purchasing from known insurers with good reputations.

Technical implications

Smart contracts are implemented on blockchain infrastructure, benefiting from the immutability characteristics of this technology. When using public blockchains, smart contracts make the rules written in the code transparent and auditable by everybody. Finally, they can also use crypto assets to settle contract terms. Among several other advantages, smart contract technology diminishes the friction among the contract participants by executing objective contracts coded in software that generates deterministic payouts, without the interference of humans. However, smart contracts also brings some challenges which include the lack of adoption, level of integration with traditional payment means, scalability and the integration with the external world outside of blockchain. Feeding data into smart contracts might seem a straightforward task but it also carries its own challenges. To understand them, it’s necessary to consider that one of the biggest motivations to use blockchain technologies is the security accomplished by using its decentralized characteristics. Therefore, the integration between the blockchain and the external world must be performed in a way that does not compromise the security achieved by the decentralization. This means that there should exist multiple data sources, and the communication channels in this interface must be securely encrypted and authenticated. Another problem related to smart contract implementation is the scalability limitations of the blockchain infrastructure. Permissionless blockchains, known to provide the best security given their level of decentralization, are also the least scalable. Such technical challenges, either on integration or scalability, pose several questions related to the architecture design of smart contract applications. This work builds on such problem analyses and proposes some designs to minimize their impact.

1. Introduction

Parametric insurance, including flight delay protection, is a compensatory mechanism that uses pre-agreed parameters to determine the outcome. The EU Flight Compensation Regulation is one example of this. As technology continues to integrate into every aspect of daily lives, this paper explores the ways in which Blockchain can be used as an oracle to collect information, parametrically determine the outcome, and execute the settlements for the agreement.

The main focus of this paper is on Parametric Flight Delay Insurance. This use case has been developed in collaboration with the Monetary Authority of Singapore (“MAS”), MIT’s Digital Currency Initiative (“DCI”), and MIT Sloan School of Management (“ MIT Sloan”).

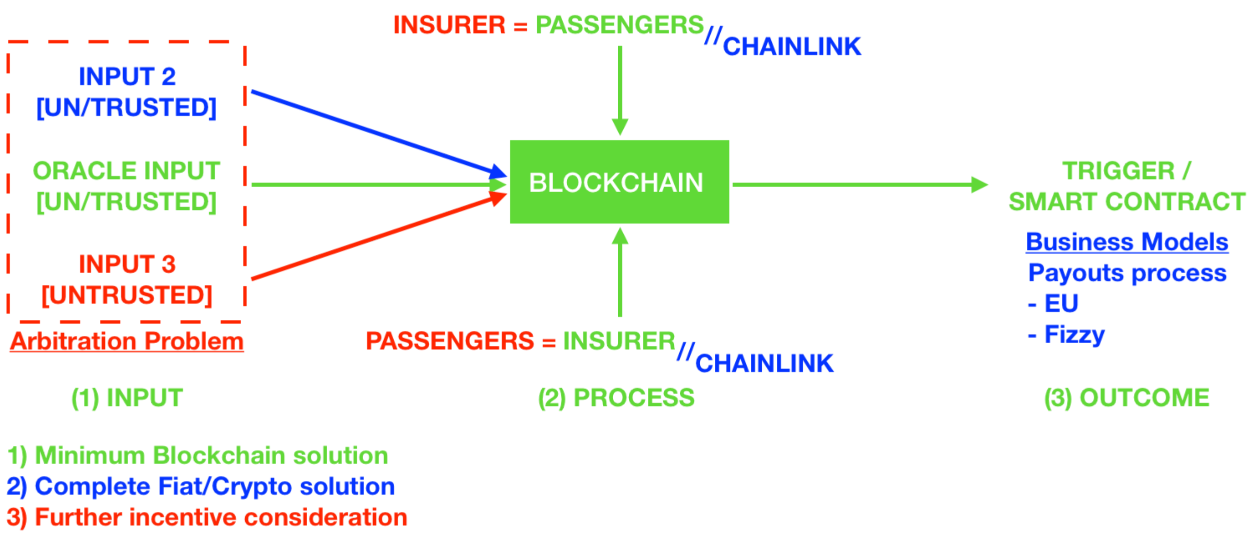
The objective of this paper is to creates a framework that applies blockchain to achieve multi-party agreement by creating a set of logic experiments to test and explore scenarios where the Oracle providers have different incentives and preferences. This paper does not provide a “right or wrong” solution, nor does it define or identify the “source of truth”.

The concept of an Oracle provider is associated to problem with the interface between the blockchain and the external world, also known as the Oracle problem. This problem comes from the decentralization nature of blockchain and its applications, such as smart contracts. Whilst data on the blockchain and those used in smart contracts are inherently secure, external data are at risk of being tampered at the source. This paper will discuss how technology can tackle this Oracle problem using technical integration and logical design of the smart contract application.

The pain points identified in a blockchain oracle can be summaries as the following:

1) What information to collect?
2) Which source to trust and which to trust more?
3) How to securely collect, analysis, and action?

What information to collect depends on the situation and is therefore not the focus of this paper. Instead to analyse 2 and 3, a generalise multiparty oracle system is broken down into 3 parts: 1) Input, 2) Process, and 3) Output.


INSURER = PASSENGERS//CHAINLINK
INPUT 2 [UN/TRUSTED]
ORACLE INPUT [UN/TRUSTED]
INPUT 3 [UNTRUSTED]
Arbitration Problem
BLOCKCHAIN
TRIGGER / SMART CONTRACT
Business Models
Payouts process
- EU
- Fizzy
PASSENGERS = INSURER//CHAINLINK
(1) INPUT
(2) PROCESS
(3) OUTCOME
1) Minimum Blockchain solution
2) Complete Fiat/Crypto solution
3) Further incentive consideration

"Truth" in this paper's use case is: whether an event has occurred.

The goal is to understand how an Oracle can be designed to fully integrate each stakeholder into a seamless compensatory system that allows market dynamics and self-selection to function without the existing drawbacks of friction and delay.

2. Literature Review

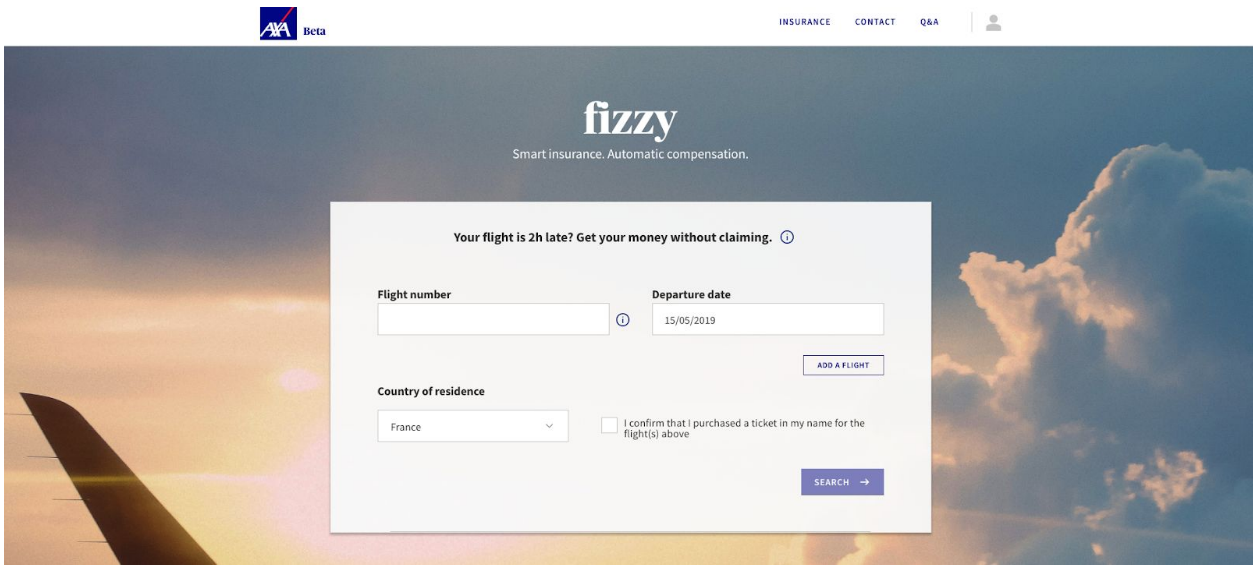
2.1. Blockchain and insurance - value in smart contracts

Insurance as an industry relies on antiquated system with many potential points where information might be lost or miscommunicated, or settlement times unnecessarily extended. A prime example is the continued reliance on paper contracts, which are prone to errors and require human supervision and typically lengthier resolution times.

Blockchain has interesting applications in parametric insurance, both on the data input side, known as the oracle, as well as the payout side. There are a few cases where blockchain is being applied in this manner.[1]

[1] CBInsights Research Briefs. "How Blockchain is Disrupting Insurance."

2.1.1. Industry examples



Your flight is 2h late? Get your money without claiming.

Flight number:

Departure date: 15/05/2019

ADD A FLIGHT

Country of residence: France

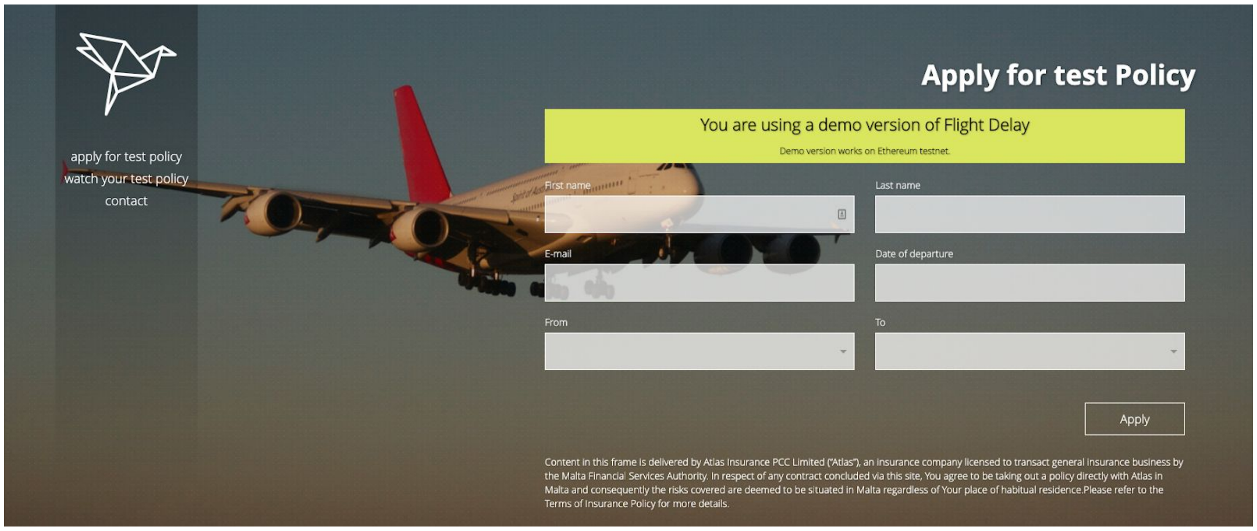
I confirm that I purchased a ticket in my name for the flight(s) above

SEARCH

AXA Fizzy

Most of these cases are pre-commercial. One of the most relevant for this project is AXA's Fizzy product. AXA is a large European Insurance company which recently launched Fizzy, a pilot flight delay insurance product that sits on the public ethereum blockchain. The value proposition here is in the smart contracts, which act as a simple triggering mechanism and also a transparent record of the contract details. AXA's stated goal is to bring more transparency into insurance thereby increasing customer trust, and overcome a common consumer feat that insurance companies will try to "trick" consumers. Currently, Fizzy issues payouts in fiat currencies, but has plans to use ether for payouts in the future.[2] They have experienced blockchain-related challenges around both around the data feeding the smart contracts, "It's a lot more complex than expected! Air traffic data is not 100% available nor clean," alongside more traditional implementation challenges, "the need to test the flight eligibility is hard to accept for potential business partners. In traditional insurance, we would insure all flights and it would be up to the customer to provide the proof of delay."[3]

[2] Higgins, Stan. "AXA Is Using Ethereum's Blockchain for a New Flight Insurance Product."
[3] Clement, Alexandre. "Fizzy.axa Smart Contracts explained."



You are using a demo version of Flight Delay

Demo version works on Ethereum testnet.

Content in this frame is delivered by Atlas Insurance PCC Limited ("Atlas"), an insurance company licensed to transact general insurance business by the Malta Financial Services Authority. In respect of any contract concluded via this site, You agree to be taking out a policy directly with Atlas in Malta and consequently the risks covered are deemed to be situated in Malta regardless of Your place of habitual residence.Please refer to the Terms of Insurance Policy for more details.

Ethersic

The other prime examples of this type of product in the market come from Ethersic. This Switzerland-based insurance platform startup had developed a range of parametric insurance products that utilize blockchain technology, including hurricane, crop, and flight delay insurance, though so far the flight delay product is the only one that has been licensed. The products also use ethereum-based smart contracts, but unlike Fizzy, can be purchased using either fiat currency or ether.[4] Ehtersic has designed their system to address issues around trust and Oracle data quality: "oracles working with Etherisc will have to stake DIP tokens as an economic incentive to provide high quality data."[5]

2.1.2. Parametric insurance

To set context, it's useful to first lay out the differences between parametric and traditional insurance products. Traditional insurance involves the insured party paying a premium in exchange for the promise an actual loss from an incident or named peril. To judge that actual loss requires and investigation and assessment, which informs how much will be paid out after the event happens. The objective is for the insured party to be "made whole."

Parametric insurance, on the other hand, revolves around a pre-agreed payout based on projected loss and probability, and is automatically triggered when a parameter is met (e.g. flight delay) based on inputs from from a trusted source. It has logical applications in the flight delay

[4] Insurance Journal. "Insurtech Startup Etherisc Offers Blockchain-Based Flight Delay Insurance"
[5] Ethersic. "Oracles: How Is Information Quality Ensured?"

use case because it can easily be purchased digitally – separately or bundled with tickets, and there are limited hassles to claim the benefits, since its paid out to traveler as soon as the delay is confirmed.

Table 1: Comparison of Traditional and Parametric Insurances[6]

	Traditional Insurance	**Parametric Insurance**
Payment Trigger	Payment triggered by actual loss of or damage to a physical asset.	Payment triggered by event occurrence exceeding parametric threshold.
Recovery	Reimbursement of actual loss	Pre-agreed payment based on event or index value
Basis Risk	**Generally smaller:** policy conditions, deductibles and exclusions	**Generally larger:** correlation of chosen index with actual event, actual payout vs. loss sustained. *Note:* Basis risk cannot be fully eliminated in parametric insurance but can be reduced using more sophisticated trigger or payout mechanisms
Claims Process	**Generally slower:** complex and based on risk adjuster assessment post-loss	**Generally faster:** Transparent and predictable because it's based on a parameter, no need for assessment, leads to quicker settlement
Term	Typically annual, less common to see multi-year deals	One-off or multi-year
Structure	Uses standard insurance contract wording so limited customization	Parametric contracts are tailored for index, payout, and client so highly customizable

[6] *SwissRE blog post "What is Parametric Insurance", Aug 2018*
https://corporatesolutions.swissre.com/insights/knowledge/what_is_parametric_insurance.html

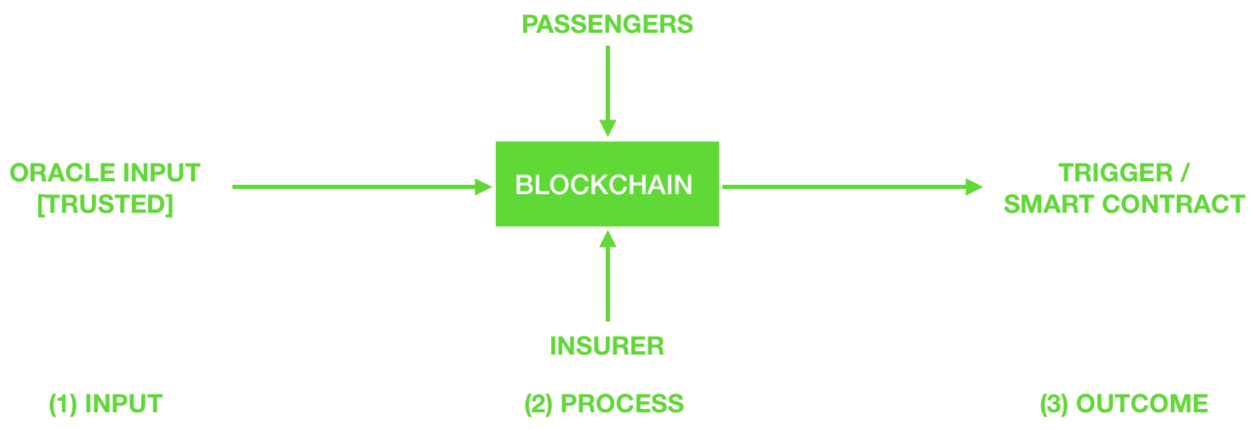
3. Key Challenges of the Oracle Problem

The Oracle problem can be divided in two parts: 1) arbitration and 2) submission. Arbitration governs the logic with which the oracle uses to make an objective decision. Submission is the technological challenge that oracle faces in securely and reliably obtaining the information.

3.1. Information arbitration

One key aspect of the Oracle problem is the arbitration - the decision process that determines the “truth” - i.e., whether an event took place. As oracle and smart contract operates objectively by following lines of computer code, it does not determine the actual truth of the matter but rather it decides that based on the instructions and formula given to it by the programmer.

3.1.1. Single source of information



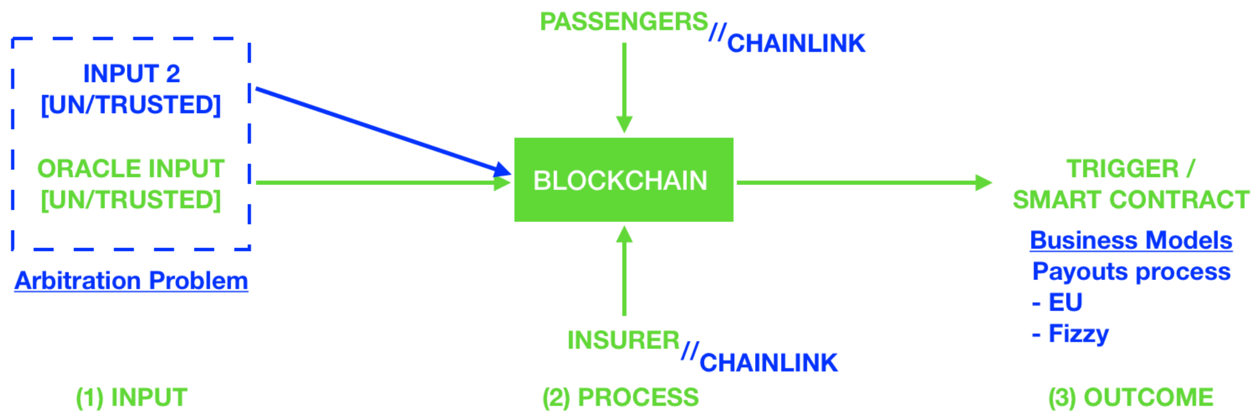
In the minimum application where there is only one information source, the arbitration problem is avoided. The Oracle system will process the information and execute the smart contract in accordance with the pre-agreed instructions.

In the absence of confirmatory source of information, for this Oracle to be trusted it implies that the one information source must be “trustworthy” and accurate from the perspective of the beneficiary of the smart contract. Note: accuracy is defined as being representative of the true state of event. Some examples are: governments agencies or trusted information providers.

The trustworthiness of a source is subjective insofar as the beneficiary is concerned. The individual is presumed to only enter this type of contract if the person is wholly satisfied with the information this one source will provide.

3.1.2. Multiple sources of trusted information

Arbitration becomes a potential problem where there are multiple sources of information and agreement is necessary. When making decisions the Oracle has no subjectivity. It is therefore necessary to develop an algorithm that will determine how the information is processed and which is to be accepted by the smart contract code.



To illustrate this issues, first consider an Oracle with two trusted sources:
Information received from both sources are considered equally trustworthy and accurate, i.e., neither is implicitly more representative of the true state of event than the other. An impasse occurs if the information are accurate but differs from one and other due to imprecision, that is the sources are accurate but a) the information are not identical, and b) the different states represented straddle a go/no go outcome according to the parameters of the smart contract.
E.g., Compensation is triggered if flight is delayed by 2:00:00hr

Sources are accurate: c.2:00:00hr but imprecise ±0:00:01

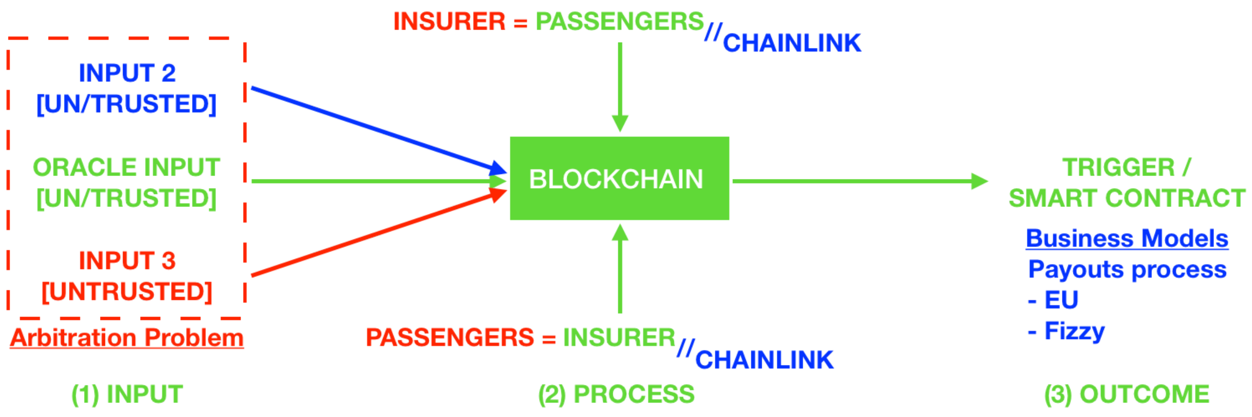
a) Source A: 2:00:01 (compensate); Source B: 2:00:02 (compensate) ⇒ compensate

b) Source A: 1:59:58 (not comp.); Source B: 1:59:59 (not comp.) ⇒ not compensate

a & b) Source A: 2:00:01 (compensate); Source B: 1:59:59 (not compensate) ⇒ **impasse**

Now consider the same issue with three or more trusted sources:
The existence of at least one contrary perspective to the majority will always require an objective arbitration. It is not immediately clear whether the majority is reflective of the true state of the world or that the minor is correct.

3.1.3. Multiple sources mixed information



The introduction of untrusted sources brings additional challenges: a) accuracy of the information, and b) incentives for foul play. The main example of untrusted sources considered in this paper is user generated information or crowdsourced data.

Firstly unlike trusted sources that are implicitly accurate, untrusted sources may or may not be accurate. It is therefore necessary to design an arbitration mechanism that will result in an agreeable state of events.

Secondly on incentive, when an individual has the ability to alter the outcome in their favour by changing the input, they are like to do so. Consider the use case of flight delay insurance:

a) assuming that every passenger will report their landing time to the oracle, irrespective of their stake in the insurance, i.e., insured or uninsured
b) the insurer take no action to steer the outcome
c) every uninsured passenger is interested in providing accurate information
d) every insured passenger is incentivised to maximise financial return by reporting a delay

as soon as one passenger enters into the insurance contract the Oracle can no longer be relied on to always provide the truth. This because the Oracle cannot return an “on time” output. Below are three possible outcomes:

1) no one is insured, accurate information is provided but no outcome is necessary
2) at least one person is insured ⇒ impasse or delayed if ture
3) everyone is insured, report of delay is certain and unanimous irrespective of the true state of event

with self interest in play and in the absence of an agreeable arbitration mechanism to resolve the impasse, 3) where everyone will obtain insurance and report a delay is the likeliest and most stable outcome.

3.1.4. Arbitration mechanism

Given the objectivity of blockchain Oracle, the onus is on the contracting parties to pre-agree an acceptable algorithm for decision making. As illustrated, where there are multiple sources of information input, impasse is possible when trusted sources disagree or when untrusted sources are incentivise to game the system with false information.

One way to break an impasse is to apply a formulaic approach. Here are some examples:

1. Majority decision
    - the simplest way to arrive at an agreement is by going with the most vote
2. Average
    - if the information is on a continuous spectrum, the aggregated data can be averaged and used as the output benchmark
3. Supermajority
    - agreement is reached when x% of the input sources agree, where x>50. The excess percentage above simple majority adds confidence to the outcome in a trusted environment
4. Weighted majority
    - where a given source is inherently more trustworthy, weighting can be assigned to differentiate the sources. The trustworthiness can be a function of technology, location, reputation, etc. e.g., from the use case, the Civil Aviation Authority of Singapore is the definitive source of information on arrival time versus a commercial information supplier, e.g., FlightStats.com, or passengers self reporting
5. Unanimous
    - unanimity can be made a necessary condition for agreement

Whilst these approaches can be used effectively to reach and enforce an agreement in a trusted environment, none addresses the incentive issues regarding untrusted sources. Two methods are proposed to address this:

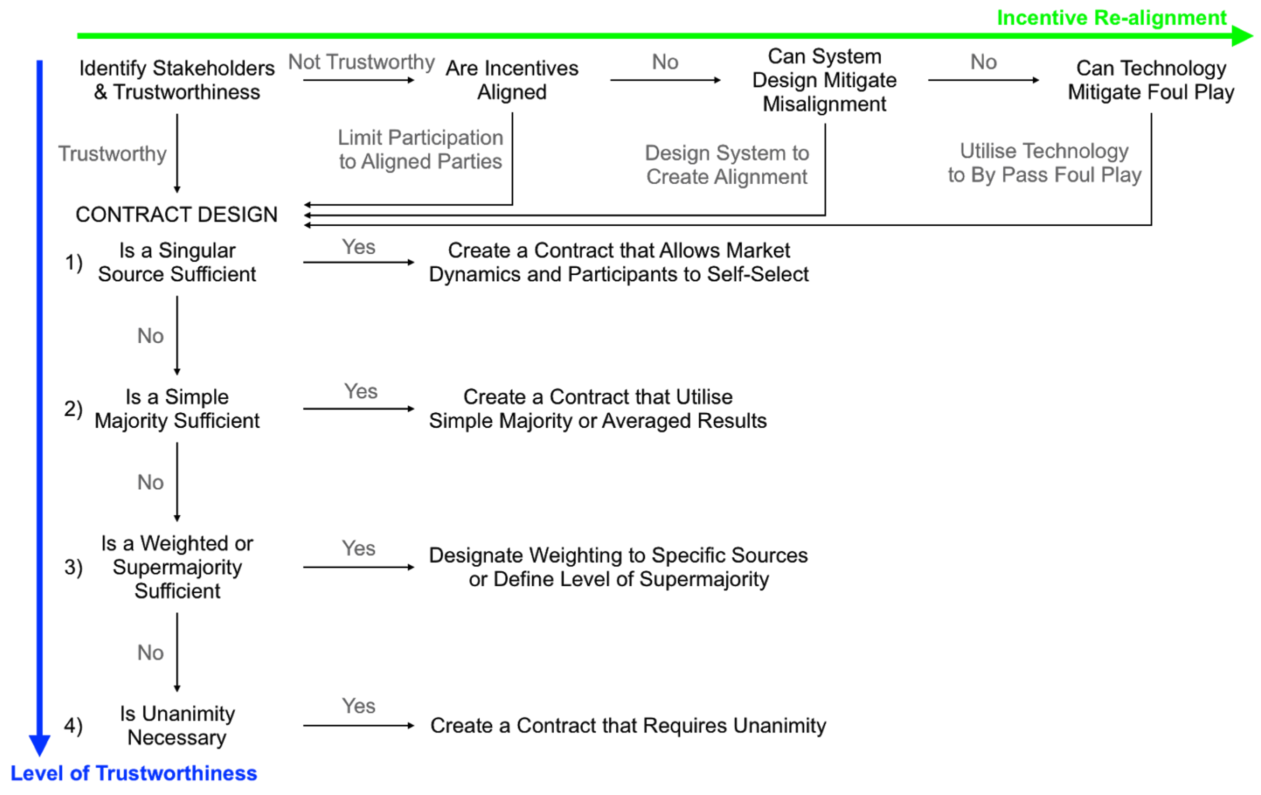
1. System design
    - By clearly identifying the different incentives within a given application, it is possible to design the contract that minimises or mitigates potential conflicts.

From our use case: a two-sided market can be created with (insured) individuals seeking to genuinely protect against delays; and (uninsured) others gambling on on-time arrival, to offset the incentives to lie by rewarding both outcomes and avoid collusion.

2. Technology
    - The growing installed base of IoT and nearly ubiquitous ownership of smartphones, spatial and temporal data can be cheaply and easily collected. By integrating technology into the data collection this opens up new opportunities. Building on the use case:
        - IoT can be implemented at the airport gate to monitor traffic over time. When passengers disembark, they create a sudden surge of traffic through the gate, and when properly calibrated this can reliably be use as a data source to indicate arrival time
        - Cell phone signal can be use to timestamp arrival as most passenger will switch on their cell phone or disable Airplane Mode on arrival. This method is particularly effective when combined with geolocation data to calibrates for anomalies such as when people switch on their phones after a delay, either out of habit or with the intention to cheat the system

3.1.5. Decision framework

Each functioning Oracle is different depending on the level of inherent trustworthiness, incentive alignment, and arbitration mechanism. Below is a proposed logical framework for consideration when creating an oracle/smart contract system:


Incentive Re-alignment
Identify Stakeholders & Trustworthiness
Not Trustworthy
Are Incentives Aligned
No
Can System Design Mitigate Misalignment
No
Can Technology Mitigate Foul Play
Trustworthy
Limit Participation to Aligned Parties
Design System to Create Alignment
Utilise Technology to By Pass Foul Play
CONTRACT DESIGN
1) Is a Singular Source Sufficient
Yes
Create a Contract that Allows Market Dynamics and Participants to Self-Select
No
2) Is a Simple Majority Sufficient
Yes
Create a Contract that Utilise Simple Majority or Averaged Results
No
3) Is a Weighted or Supermajority Sufficient
Yes
Designate Weighting to Specific Sources or Define Level of Supermajority
No
4) Is Unanimity Necessary
Yes
Create a Contract that Requires Unanimity
Level of Trustworthiness

3.2. Information submission

The submission problem is composed of the challenges involved in capturing information from potentially multiple data sources and feeding it into blockchain. This part of the Oracle problem typically involves technological problems such as the encryption and signature of data sent by the Oracles. Another problem is the decentralization of the datasources. The Ethereum platform offers some basic tools to handle this problem but more security might be needed particularly if there are multiple data sources and a low level of security on each of them. Other platforms offer services in the market to overcome those issues - Chainlink is one of those. This platform provides encrypted and secure mechanism to convey data from the data sources to the Ethereum smart contract, using SGX chip technology from Intel. This technology provides a hardware secure environment to execute software code. This security mechanism relies on the trust of the participants on the technology developed by Intel.

4. Smart Contracts

4.1. Cryptocurrency and blockchain overview

Crypto currencies and blockchain have been a hot topic in the past 4 years. This story started in 2009 when a person or a group of people under the the pseudonym of Satoshi Nakamoto, published the seminal paper, Bitcoin - A Peer-to-Peer Electronic Cash System[7]. A couple of months later, they launched the first net with the implementation of this protocol. The original goal of this system was to implement a currency that it is not dependent on trusted third parties to address the double spending problem. The main component of this system architecture is the blockchain, which allows every participant of the network to check and audit the system by inspecting the public ledger composed of blocks of transactions. This is a completely different paradigm in which the security is based on the decentralization of the verification task, resulting in incredible system robustness. These characteristics are accomplished by having an intricate consensus mechanism in which miners generate blocks that define the sequence of transactions and verifiers (or full nodes) who validate the compliance of the miners' work according to rules coded in software[8]. One specific miner is compensated with crypto coins for their computational effort to generate blocks in a race alongside other miners. On the other side of this consensus game, full nodes who are not interested in having their coins inflated, follow the rules written in software implementation while validating the block transactions. This incentive structure creates a "mutual assured destruction" setting in which miners and full nodes lose if either decide to abandon the consensus or software rules in their own benefit. Therefore, to change the rules it is necessary to convince almost all network members, including miners and full nodes, that the software must be updated. This is the intricate but robust consensus mechanism achieved in a sufficient distributed blockchain-based network. It is interesting to note that the more distributed it is, meaning more participants, the more decentralized the network is. The increase of the decentralization makes any change in the consensus protocol more difficult, given the need to convince all players in the network to accept them. As a result, the network gets more robust with size in the sense that transaction history will not change

[7] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System."
[8] Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger."

easily. These conclusions are explored by the several applications, smart contracts being one of those, implemented on top of the blockchain technologies.

4.2. Smart contracts overview

Contracts in real life are agreements created between agents, individuals or legal entities, that engage in some sort of interactions that in most of cases involves transfer of assets as a compensation for services or products exchange. Contracts are formalized by documents that describe the conditions in which services and products should be transferred and how assets should be sent and received by the parts involved. Well-defined contracts accurately describe all conditions and possible scenarios in which transaction can occur and, moreover, they also do not leave any ambiguity regarding to the contract settlement. It is interesting to note the resemblance of a well-defined contract and a well-written software code. The aim for software implementation is to generate a code that collects inputs and processes them to generate outputs in a deterministic way, handling all types of inputs properly and producing coherent outputs according to intent of the software developer. In fact, legal language is designed to systematize human actions and conditions to provide deterministic interpretations. This parallel was first realized by Nick Szabo, who has a background in law and computer, in his paper "Formalizing and Securing Relationships on Public Networks"[9]. In this paper, the author explores the possibility of using software, networks and cryptography to formalize and secure transactional relationships between parties who decided to enter in a contract agreement. The benefits of such systems are quite clear and those include objective clause interpretation rather than subjective human and manual processing, and also automated contract settlement. At this point, it would be hard to implement the ideas from this paper without using a trusted third party. Moreover, since financial assets such as banks are organized in closed loop consortia and under strong regulation, it was challenging to integrate cryptographically secure software contracts with traditional payment systems.

This scenario began to change with the emergence of cryptocurrencies and blockchain technologies. The ability to transfer crypto assets without the need of a trusted third party triggered an interest in developing infrastructure to implement smart contracts. In addition, the immutability and decentralized characteristics of blockchain technology are highly suitable for

[9] Szabo, Nick. “Formalizing and securing relationships on public networks.”

implementing such transactions agreements with transparency and security for all parts involved. Consensus mechanisms play a important role in the underlying foundation of a smart contract since one of the most important aspects of a contract is the fact that it registers the mutual agreement beforehand and prevents participants from changing it to benefit themselves as time goes by and information about the future becomes available. That is the reason that contracts are signed and in some cases registered by a notary. With blockchain technology, smart contracts benefit from an open and public notary enforced by the decentralization of the network.

There is a long list of benefits from using smart contracts. First, transactions are more transparent since contract code is publicly stored in the blockchain. Second, contract settlement is processed automatically, avoiding cumbersome paperwork procedures and painful claim processes, particularly when there is an asymmetry on the financial execution capacity of the contract. Since smart contracts have rules coded in software, contract execution is performed in a objective way according to the rules agreed beforehand. Finally, smart contracts can benefit from the cryptocurrency implementation to command automated asset transfers as a means of compensation for contract participants as conditions are met. This provides completeness to such implementation and makes the use case for smart contracts on top of cryptocurrencies and blockchain technologies very attractive.

On the other hand, smart contracts bring a comparably long list of challenges. Mass adoption, network effects, underlying technology issues, and security, among others, are the root of various obstacles that get in the way of broad usage of smart contract technology. Below is a non-exhaustive list of the main challenges of smart contract implementation:

1. Contracts in code can be difficult to implement. Certain conditions that are easy to understand and process using "conventional human sense" can be difficult to translate into hard lines of program code.
2. Cryptocurrencies are not stable and the vast majority of people prefers to deal with traditional payment means, e.g. fiat currency.
3. Cryptocurrencies are not well understood and in most geographies are not properly regulated. This means users are not incentivized to adopt them.

4. The decentralized consensus mechanism does not scale very well.[10] To give a sense of this issue, the current Bitcoin mainnet is able to process on average, 7 transactions per second. Conversely, traditional credit cart operator networks can process more than 50,000 transactions per second. The process to determine the next block - mining - and the verification mechanism are intense involve high levels of and processing capacity and network communication. Therefore, highly decentralized networks tend to not scale well.
5. Interface between blockchain infrastructure and real world can be tricky. This problem is a important problem and it is one of the motivations of this paper. It is often referred to as the Oracle problem and it involves security and design aspects. We will double click on this problem in the following sections.

In spite of challenges mentioned above, smart contracts promise to generate value to users, achieved through the evolution of technology and design implementations that circumvents the obstacles of deploying smart contracts on the blockchain infrastructure.

4.3. Oracle problem

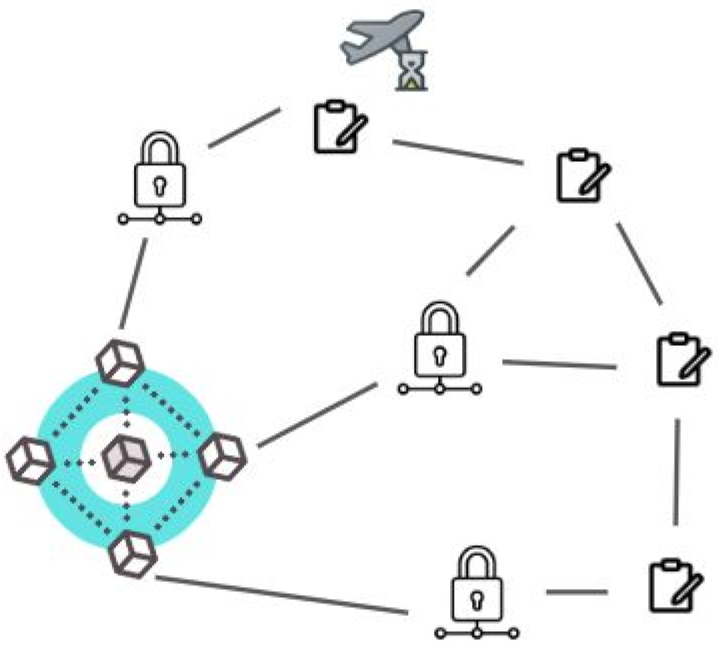
At this point, we will dive into smart contracts at a higher level of detail. It was previously mentioned that they are pieces of code that initiate software as form of real world contracts. Where is that software executed? To answer that question, it is necessary to add more information about the underlying technology of smart contracts: the blockchain. When miners are creating blocks with transactions, they are creating a sequential and historical list of data that consists of a chain of blocks. All miners and full nodes, or verifiers, posses a copy of this chain of blocks that was validated and agreed to be the single source of truth by the consensus mechanism. In simple terms, it is reasonable to think of the blockchain as a group of computers that are executing a consensus software on top of a replicated database composed by a chain of append-only databases. It is on top of this infrastructure that smart contracts are deployed. The script of the contract and the data used by the script are stored in this replicated chain of blocks. Whenever the smart contract must be executed, during the mining process, the script software is read from the local chain, the code is run and finally the output of this execution is stored on the next block to be appended on the chain. It is important to mention that the

[10] Anamika Chauhan et al. “Blockchain and Scalability.”

outcome of this contract execution will be replicated to all other nodes of the network. In this process, it is important to understand which node will be responsible for the contract execution. In reality, multiple nodes will execute this contract. Given the decentralized nature of the blockchain, several miners will compete to generate the next block, meaning that if a smart contract needs to be executed on the next block, several mining nodes will execute it in hopes that they will win the race for producing the next accepted block. Therefore, the script of the smart contract must output the same results even though it’s being executed by different miners asynchronously: this means that the smart contract execution must be decentralized and consistent. Considering that all miners have the same copy of the blockchain and that the smart contracts will collect data only from the blockchain, reaching consistency is not a very challenging goal. However, an ability to integrate with the world outside blockchain is a basic requirement to implement almost all real world smart contracts. This integration generates some problems. Let's list the most important ones:

- **Data integrity -** Unlike blockchain data, external chain data inputs do not have their consistency and immutability guaranteed by the consensus mechanism. This means that multiple executions could potentially output different results.
- **Security -** Since smart contracts applications command funds or asset transfers, the integration between the real world and the blockchain must be tamper proof so that malicious attacker are not able to change data that smart contracts are using to execute in the blockchain.
- **Datasource decentralization -** Given the level of security generated by the decentralization, smart contracts are expected to provide the same security standard end to end. This means that users of a decentralized smart contract application, in short DApp, are not willing to compromise the level of security achieved by this configuration by integrating with a single external data source that might be vulnerable to tampering.

Those are the challenges of the constitute the "Oracle problem". Let's consider the example of a smart contract application for flight delay insurance (“FDI”), the use case raised in Section 3. This contract will compensate passengers, who paid a premium fee, in the case that their flight is delayed. Conversely, this contract should return the funds to the insurer in the case that the flight is not delayed. There are some important questions in this application related to the oracle problem: Who will inform the contract that the flight is delayed? How will this information be conveyed into the blockchain while still preserving the security premisses? The figure below shows a diagram of the oracle problem. Each block represents a different miner node that could potentially start the FDI execution - eventually each of these nodes will connect to the various flight delay datasources.



Oracle problem at FDI DApp

5. Smart Contract Architecture Design

5.1. Architecture design

Given the scalability challenges and oracle problem discussed in the previous sections, the architecture design for a smart contract application is a critical consideration. In this project, we explored two types of architectures for smart contracts which are listed below:

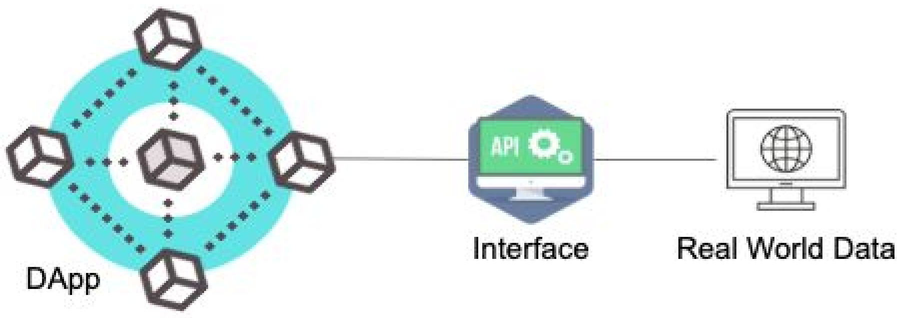
1. **On-chain -** This architecture is characterized by the deployment of the smart contract script as part of the blockchain data, meaning its execution will be done by the miner nodes. Ethereum is a blockchain infrastructure that provides comprehensive support to implement smart contracts and is the preferred choice for DApp developers.[11]
2. **Off-chain -** It is also possible to implement smart contract code outside the blockchain infrastructure. The main concept of this architecture is to separate the processing and

[11] Vujičić, Dejan et al. “Blockchain technology, bitcoin, and Ethereum: A brief overview.”

verification parts of a smart contract. The processing part is executed outside the chain while verification is performed on chain through the validation of signatures and pre-built transactions. This idea was created by Thaddeus Dryja at the MIT Digital Currency Initiative.[12]

5.2. Ethereum

Ethereum is well-known blockchain infrastructure characterized in particular by its smart contract support. This platform offers a vast set of tools to develop and deploy smart contracts on top of its native currency token, the Ether. In fact, Ethereum offers a Turing complete programming language to develop smart contract code which also includes an API that allow interface software to call contract functions inside the Ethereum blockchain. This comes in handy to address the consistency part of the Oracle problem. The figure below shows a diagram with the representation of such interfaces.



Ethereum external interface diagram

Given the resources available on Ethereum to develop smart contracts, very complex logic can be implemented. This is an advantage since real world contracts are rich in clauses and complexity. However, this introduces two problems. The first is associated with the scalability issues of the blockchain. The comprehensive nature of the Ethereum platform amplifies the performance limitations of this technology. The second problem is related to cost of the execution of the smart contract. As discussed previously, miners are compensated by the their work on block generations. Since smart contracts in Ethereum are executed on-chain, this

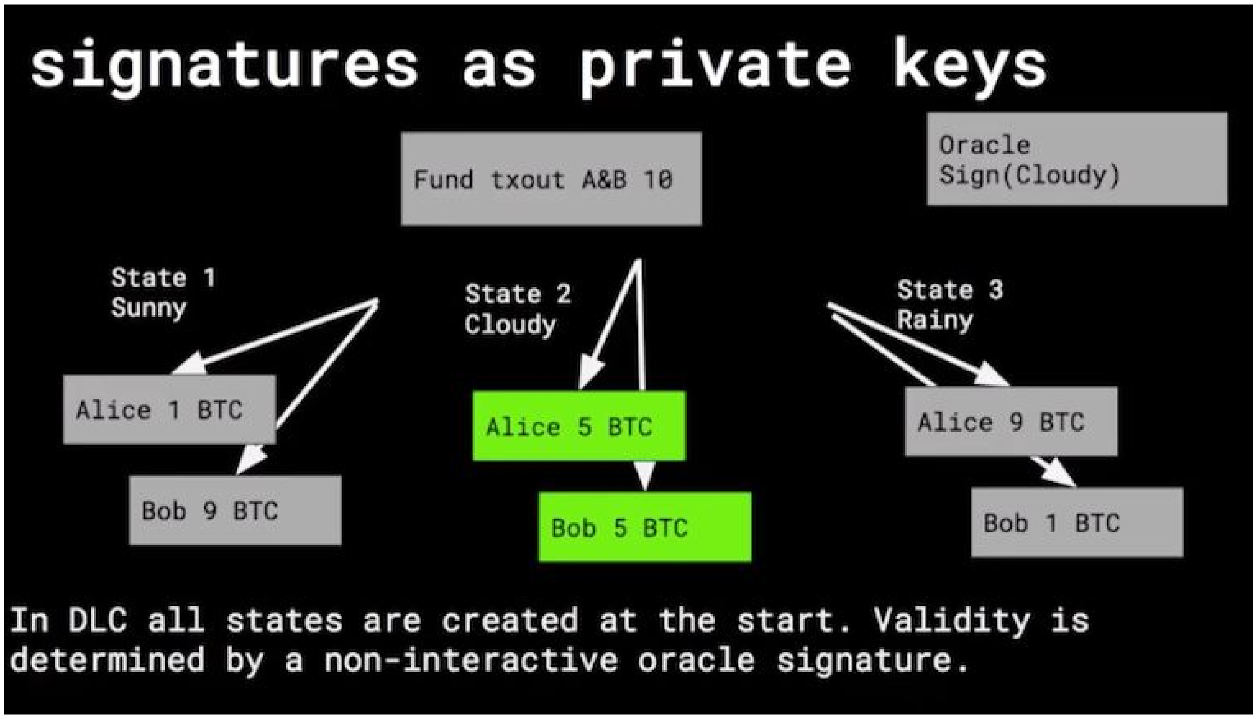
[12] Dryja, Thaddeus. “Discreet Log Contracts.”

means that miners will have to execute this code and tie it to the next block generation. This process is paid and it is measured in Gas which is a subunit of Ether.

5.3. Discreet Log Contracts

An alternative approach to execute smart contacts is the Discreet Log Contracts, in short DLC.[13] The central idea of DLC is to compute ahead of time all the possible transaction outputs of a contract and distribute those transactions templates to the contract participants. Let's analyse one contract example illustrated in the figure belows, extracted from one Thaddeus Dryja's DLC presentations. This contract is a fictitious betting between Alice and Bob about the weather in the future. There are three possible outcomes which are sunny, cloudy and rainy. An Oracle is responsible for providing the weather information on the betting day. The figure shows the payouts depending on each Oracle outcome. The core construction of DLC uses Schnorr signatures in its signature scheme. The construction explores the fact that Alice and Bob can compute only the signature of the set of transactions that correspondent to Oracle information output, making only this specific transaction broadcastable and spendable. As security provision, in case Alice or Bob tries to cheat by sending the wrong transaction, the other part can immediately transfer all funds to himself or herself using a revocation key. The full details can be found in the reference cited.[14]

[13] Dryja, Thaddeus. “Discreet Log Contracts.”
[14] Dryja, Thaddeus. “Discreet Log Contracts.”



Discreet Log Contract scheme diagram - Figure from Thaddeus Dryja presentation

This creative mechanism allows the implementation of the contract logic to be done outside the blockchain and therefore not constrained by the restrictions of this environment. The drawback of DLC is the fact that it is necessary to pre-compute all possible outputs beforehand in a discreet manner. Moreover, in the case that a contract generates multiples output possibilities potentially because of multiple interactions, the transaction anticipation can become cumbersome. The limitation here is not the amount of pre-computed transactions, since these calculations are performed off-chain, but the complexity of your contract scheme.

5.4. Other blockchain technologies

Nowadays there is an explosion of blockchain technologies. It seems that almost daily there is a new platform with new features, promising to solve Bitcoin and Ethereum scalability issues while maintaining their level of security and decentralization. Most notably, permissioned blockchains became quite popular in the corporate world particularly in same industry companies organized in a consortiums. Permissioned blockchains require permission to participate and therefore they can use technologies and constructions that scale very well. However, this benefit comes at cost of decentralization and a corresponding increase in the risk of the blockchain being tampered with or censored. Some permissioned blockchains, such as R3-corda, were simplified to the extent that the own providers admit that they are no longer blockchains in the classical sense.

Other public projects, such as Algorand, present very interesting cryptographic technologies but have not reached the level of network distribution to be considered safe from the decentralization perspective. Those projects might succeed in the future and maybe will displace Bitcoin and Ethereum as top 2 biggest blockchains, but this is not the current reality.

Given the fact that permissioned blockchain are specific and perhaps not suitable to mass market applications, and the fact that other blockchain projects are still incipient, this project concentrated in analysing only Bitcoin and Ethereum blockchains.

5.5. On-chain versus Off-chain comparison

This session presents a summary table with the pros and cons of DLC and Ethereum smart contracts.

Technology	**Pros**	**Cons**
DLC	● Scalable ● Better performance ● Cheaper to operate	● Hard to develop ● Complex
Ethereum	● Easy to develop ● Allows highly complex logic	● More expensive due to cost paid to miners ● Does not scale well

6. Proof of concept

6.1. Ethereum Implementation

As part of this project, our team developed a proof of concept for a flight delay insurance application, FDI. This prototype was developed in Ethereum and uses a Javascript web application to provide the interfaces for the three main stakeholders of this application: passengers, insurance providers, and Oracles. Passengers can access this application using either desktop or mobile internet browser. The interface between the web application and Ethereum blockchain is performed by crypto wallets - we used Metamask for the desktop and Coinbase for mobile. This application implements the workflow of the parametric flight delay insurance, as described in Section 3. All fund transfers occur on the Rinkeby Ethereum testnet. This testnet uses test Ether but the environment is very similar to the one of the actual main net.

The main goal of this implementation is to discover and explore what is involved in implementing a smart contract application. Basically, we implemented a smart contract in solidity, which is the Ethereum smart contract programming language, and this prototype followed the on-chain approach described earlier. It also incorporated digital signatures on the Oracle messages to increase the security of the Oracle interface.

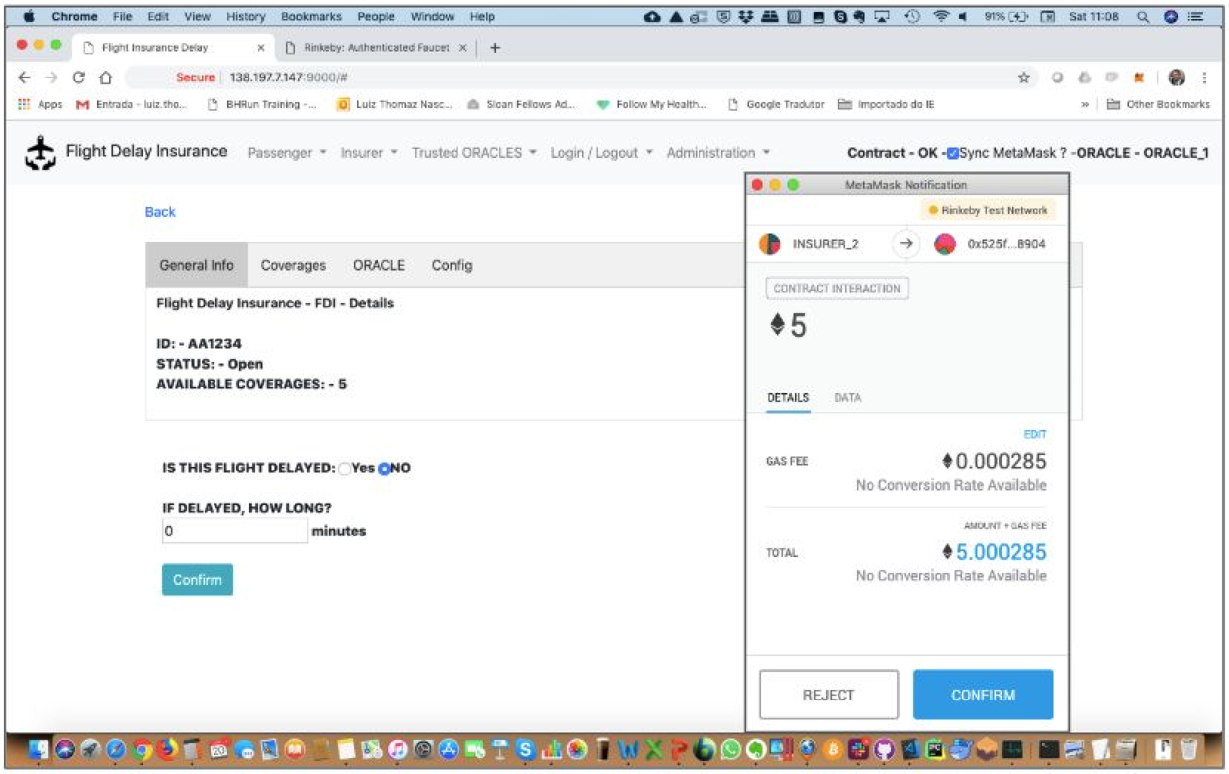
6.2. Results

The implementation of the parametric insurance application covers the basic workflow involved in this use case. There are essentially three stakeholders, who are the passenger, insurance provider and oracles. In a real world application, an oracle should not be an user but an api integration or a system. However, for the purposes of this prototype, this stakeholder was implemented as a system user. The workflow of the application consists on the following basic steps:

- Insurance provider creates a flight delay insurance coverage product.
- Passengers can navigate through a list of available coverage products, select one flight coverage and by the insurance using Ether crypto coin as payment means. This operation can be done either on desktop or mobile.

- One or more oracles can input flight delay information. Each oracle input is computed internally by the smart contract and when the total number of oracles are done with the information feed, the payouts are triggered. Either to send all contract funds to the insurer in case that there is no flight delay or to indemnify passengers using the payout parameters of the contract.
- All payouts are processed and the contract is finalized.

Below are some screenshots of this proof of concept.



Contract - OK - Sync MetaMask ? - ORACLE - ORACLE_1

Flight Delay Insurance - FDI - Details

ID: - AA1234
STATUS: - Open
AVAILABLE COVERAGES: - 5

IS THIS FLIGHT DELAYED: Yes [✓ NO]

IF DELAYED, HOW LONG?
0 minutes

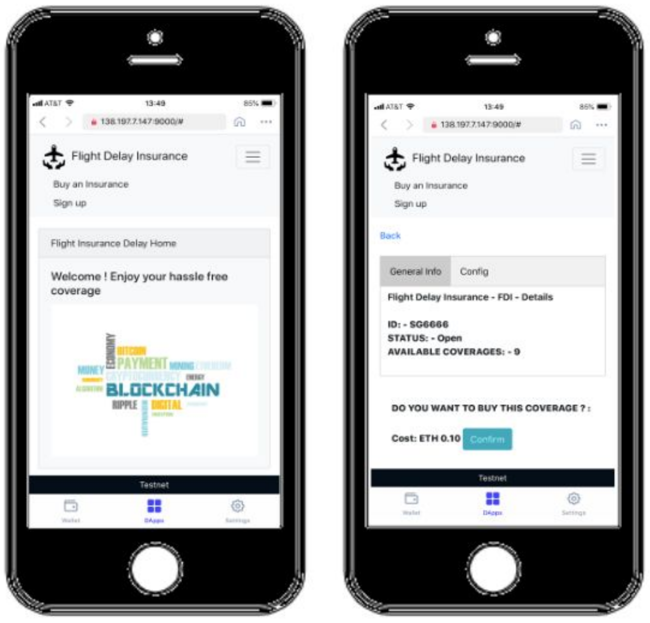
INSURER_2 → 0x525f...8904

5

GAS FEE 0.000285
No Conversion Rate Available

AMOUNT + GAS FEE
TOTAL 5.000285
No Conversion Rate Available

Desktop application



Welcome ! Enjoy your hassle free coverage

Flight Delay Insurance - FDI - Details

ID: - SG6666
STATUS: - Open
AVAILABLE COVERAGES: - 9

DO YOU WANT TO BUY THIS COVERAGE ? :

Cost: ETH 0.10

Mobile application

7. Potential Benefits and Uses Cases

7.1. Parallel contracts

Building on the information arbitration discussion, in the case where there is only one source of information, it can be argued that government agency is generally trustworthy. However, as it is not inconceivable that some individuals will consider certain governments less reliable than commercial information provider. With blockchain and smart contracts, one solution is for insurers to set up parallel contracts using different Oracle to determine the outcome and let market forces and self selection decide what is more acceptable for the insured.

7.2. Cooperative self insurance

One key significant advantage of blockchain is its multi party consensus and read/write permission. Farming cooperative is a well study and implement insurance scheme that allows many farmer in the same region to self-insure as a collective. This differs from the current use case by the fact that the insurer and insured can be the same. As addressed in the information arbitration section, this offers a unique test case on how system design and technology can influence the design of the Oracle system and insurance terms.

8. Conclusion

Blockchain and Oracles have the potential to transform the world, where multiple parties are involved in agreeing and settling a transaction that are based on reportable parameters which define a state of the world. More specifically, this paper has demonstrated how the technology can be implemented to bring together different sources of information and stakeholders to create a compensation scheme that eliminates the drawback of friction and delays.

Insofar as it is possible, there are logical and technical challenges in designing and implementing such a system. As this technology does not offer subjectivity, arbitration algorithms, system design, and integration with IoT and other technologies are necessary to address the decision impasse and incentive issues. A decision framework is proposed to aid the designing of an Oracle system given a scenario of varying trust and incentive alignment.

8.1. Further research

Further research on understanding the the impact of "instantaneity" on trust and incentive would add great value, especially to commercial entities. Here are two examples to consider:

1) Existing insurance generally have a 24-hour "Waiting Period" to avoid claimants signing up knowingly cheating the system and making a claim. Building on the use case, with many airlines now offering in-flight internet connectivity, how should the oracle system respond to an insurance purchase by a passenger that is on a flight that is already delayed? How much additional data points do the oracle need to avoid this issue?
2) Current parametric flight delay insurance uses a set of arbitrary cut-offs to determine compensation payment. This incentivises the airline to "beat the clock" by trying to reach the destination minutes earlier than each cut-off to save cost without necessarily making best effort to make up for lost time. One reason for this limitation could be the enormous resource required for each bespoke compensation if a formulaic approach is applied widely. This is a trivial problem for smart contract. How will the application of smart contract change the incentive for a more punctual transportation network?

9. References

Anamika Chauhan et al. “Blockchain and Scalability.” 2018 IEEE International Conference on Software Quality, Reliability and Security Companion

Blocksplain. “Blockchain speeds & the scalability debate.” Feb 28, 2018. https://blocksplain.com/2018/02/28/transaction-speeds/

CBInsights Research Briefs. “How Blockchain is Disrupting Insurance.” Jan 10, 2019. https://www.cbinsights.com/research/blockchain-insurance-disruption/

Clement, Alexandre. “Fizzy.axa Smart Contracts explained.” Medium. June 13, 2018. https://medium.com/@humanGamepad/fizzy-axa-smart-contract-explaind-740df52894fd

Vujičić, Dejan et al. “Blockchain technology, bitcoin, and Ethereum: A brief overview.” 17th International Symposium INFOTEH-JAHORINA, 21-23 March 2018. https://www.researchgate.net/publication/324791073_Blockchain_technology_bitcoin_and_Ethereum_A_brief_overview

Ethersic. “Oracles: How Is Information Quality Ensured?” Medium. Dec 12, 2018. https://blog.etherisc.com/oracles-how-is-information-quality-ensured-646f2f41f605

Higgins, Stan. “AXA Is Using Ethereum’s Blockchain for a New Flight Insurance Product.“ Coindesk. Sept 13, 2017. https://www.coindesk.com/axa-using-ethereums-blockchain-new-flight-insurance-product

Insurance Journal. “Insurtech Startup Etherisc Offers Blockchain-Based Flight Delay Insurance“ Oct 30 2017. https://www.insurancejournal.com/news/international/2017/10/30/469647.htm

Szabo, Nick. “Formalizing and securing relationships on public networks.” First Monday, 2(9), 1997. https://ojphi.org/ojs/index.php/fm/article/view/548/469

Satoshi Nakamoto. “Bitcoin: A Peer-to-Peer Electronic Cash System.” 2009. https://bitcoin.org/bitcoin.pdf

Dryja, Thaddeus. “Discreet Log Contracts.” MIT Digital Currency Initiative. May 1 2018. https://static1.squarespace.com/static/59aae5e9a803bb10bedeb03e/t/5a85cf21e4966bb735a9f757/1518718768309/discrete+log+contracts.pdf

Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." Ethereum Project Yellow Paper. 2014. https://ethereum.github.io/yellowpaper/paper.pdf