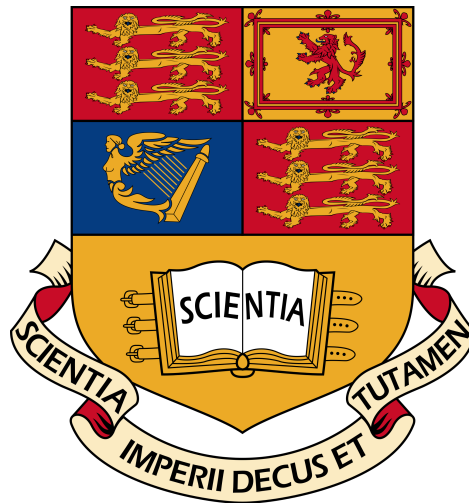


The Value Chain

Applied to the UK's Independent Film Industry



Alan Vey

Department of Computing
Imperial College London

This report is submitted for the degree of
Masters of Engineering in Computing (Artificial Intelligence)

June 2016

Acknowledgements

- Professor William Knottenbelt, my supervisor, and others in the Center for Cryptocurrency Research and Engineering, for your guidance throughout my project. It has been a pleasure working with you and I hope we will continue to do so.
- Professor Kin Leung, my second supervisor, for your advice from the beginning.
- Gerard O'Mally, and others at The Film Network, for their expertise and assistance in understanding the film industry. I hope to continue working with you to bring this solution to film makers.
- Annika Monari, for her help and support throughout all aspects of this project.

Abstract

We present The Value Chain, the first application that enables the secure distribution of digital assets and management of the associated permissions and payments, without a central party. Users' distribution agreements provide a transparent audit trail by making use of the Ethereum Blockchain.

This solution has been applied to the UK's independent film industry in an attempt to help film makers keep their films secure during distribution, and provide a more fluid management of viewing rights. With our solution, they should be able to reach larger target audiences with greater profit margins. The solution can be applied to any industry distributing a digital asset such as music or health records.

Our unique and novel approach has been tested and shown to various industry experts including The Film Network, The British Academy of Film and Television Arts (BAFTA), Innovate UK, The UK's chief science adviser Sir Mark Walport, Digital Catapult and Goldman Sachs. Feedback obtained from industry alongside our analysis has show our solution to be secure and in demand. We will be creating a company to further develop this project.

Table of contents

1	Introduction	1
1.1	Motivation	1
1.2	Objectives	2
1.3	Contribution	2
2	Background	5
2.1	Blockchain and Beyond	5
2.1.1	Overview	5
2.1.2	Cryptography	6
2.1.3	Bitcoin	7
2.2	Smart Contract	11
2.2.1	Overview	11
2.2.2	Codius	11
2.2.3	RootStock	12
2.2.4	Counterparty	13
2.2.5	Ethereum	14
2.2.6	Discussion	21
2.3	Decentralized Storage	23
2.3.1	Overview	23
2.3.2	Sia	23
2.3.3	Maidsafe	24
2.3.4	Storj	25
2.3.5	Inter Planetary File System (IPFS)	26
2.3.6	Discussion	27
2.4	Regulatory Concerns	27
2.5	The UK Independent Film Industry	29
2.5.1	Overview	29
2.5.2	Distribution Process	29

2.5.3	Security	30
2.5.4	Film Industry	30
2.5.5	Discussion	31
2.6	Related Projects	33
2.6.1	Overview	33
2.6.2	Bittunes	33
2.6.3	FilmFund.io	33
2.6.4	UjoMusic	33
2.6.5	Digital Catapult	34
2.6.6	Mediachain	34
2.6.7	Discussion	35
3	System Design	37
3.1	Overview	37
3.2	Smart Distribution Chain	37
3.2.1	Creating Distribution Contracts	38
3.2.2	Coming to an Agreement	40
3.2.3	Signing Contracts	42
3.2.4	Management	43
3.3	Hierarchical Entities	45
3.3.1	Structure	45
3.3.2	Decentralized Decision Making	47
3.4	Secure Storage of Digital Assets	48
3.4.1	Centralized	48
3.4.2	Decentralized	49
3.5	Putting it all together	50
4	The Value Chain	53
4.1	Overview	53
4.2	The Film Maker	54
4.3	The Film Distributor	60
4.4	The Film Consumer	65
5	Implementation	67
5.1	Overview	67
5.1.1	Architecture	67
5.2	Smart Contracts	68

5.2.1	Infrastructure	68
5.2.2	Contracts on Ethereum	71
5.2.3	Entity	72
5.2.4	Asset and Permission	73
5.2.5	Testing	74
5.3	Decentralized Application (DApp)	76
5.3.1	Infrastructure	76
5.3.2	Integration with IPFS	78
5.3.3	Integration with Ethereum	79
5.3.4	Testing	80
5.4	Private Server API	80
5.4.1	Infrastructure	80
5.4.2	API	82
5.4.3	En/Decrypting Assets	83
5.4.4	Database	83
5.4.5	Testing	86
6	Evaluation	87
6.1	Overview	87
6.2	Industry Response	88
6.2.1	The Film Network	88
6.2.2	Innovate UK Application	89
6.2.3	British Academy of Film and Television Arts	90
6.2.4	Digital Catapult	90
6.2.5	Government Round Table	91
6.2.6	Ujo Music	91
6.2.7	Goldman Sachs	92
6.3	Smart Contracts	92
6.3.1	Ethereum	92
6.3.2	Contract Optimization	93
6.4	Private Server	94
6.4.1	Asset Storage	95
6.4.2	Threat Analysis	95
6.4.3	Security and Testing	96
6.5	Decentralized Application (DApp)	96
6.5.1	User Experience Feedback	96
6.5.2	Testing	102

6.6	Strengths and Weaknesses	102
7	Conclusion	105
7.1	Lessons Learned	105
7.2	Future Work	106
	References	109
	Appendix A Innovate UK Feedback	117
	Appendix B User Interface Mock-ups	123

Chapter 1

Introduction

1.1 Motivation

The motivation for this project stems from the new possibilities that have arisen from advances in Technology, specifically the ideas around Bitcoin and the Blockchain, enabling the transfer of value digitally, rather than just the duplication of an item provided by previous peer-to-peer platforms.

We are trying to solve the problem of efficient, secure and transparent distribution of an item of value (an asset), without a trusted middle man and the management of the associated permissions and payments.

We will be applying this solution to the United Kingdom's Independent Film Industry to aid independent film makers who are having difficulties protecting their Intellectual Property, reaching a market in the existing structure, and making a profit from their films.

Many problems have arisen out of the world becoming more dependent on technology. Bitcoin and the Blockchain have given us new tools in dealing with some of these problems, specifically to do with transferring value online, but have introduced others:

- How do we exchange value between parties according to some agreement in a transparent yet secure way without a middle man?
- Can we use advances in technology to create more efficient Economies and better models of governance?
- How do we enable more transparency so that we may regulate and audit services better to avoid criminal activity?
- Does your data belong to the company whose service you are using (e.g. Does Facebook own your profile?, Can you regulate what they do with it?)?

The answer to any of these questions is far from trivial. On one hand, we need to create solutions that address these problems and ensure that they themselves are not lacking in security and transparency. On the other hand, we have to persuade regulators and existing businesses to allow the current policy and systems to change. Since it is rare to find a solution without trade offs, we need to devote resources to research the weaknesses of new and untested models.

1.2 Objectives

We are attempting to address some of the problems mentioned above by creating a web application that stores a structured collection of digital items of value and securely regulates the access to them without a trusted third party. The app governs the flows of value between the users creating, distributing and consuming the collection, creating transparent records in the process.

We are working with The Film Network (TFN) and the British Academy of Film and Television Arts (BAFTA) to apply our solution to the United Kingdom's independent film industry. Each of the components of a film is stored securely (i.e. soundtrack, script, the film itself, etc.) and access to them is regulated by the system. The system will facilitate the negotiation and creation of distribution contracts forming a hierarchical chain and handle all resulting payments from the consumer, through the chain, back to the creators and rights holders.

1.3 Contribution

The following are the main contributions of this project:

- A means of creating, negotiating and digitally signing smart contracts that will manage the distribution of an underlying asset and the associated payments.
- A means of expressing complex hierarchical organizational structure and democratic voting to fairly and transparently enable decisions e.g about what distribution contracts to sign or how payments will be split between individual users (shareholders).
- A means of securely storing and efficiently retrieving a digital asset with no single point of failure and no server down time.
- An Application Programming Interface (API) enabling other applications to be build using our solutions.

- A decentralized application demonstrating the use of these features in the UK's independent film industry.

Chapter 2

Background

This section will introduce the concepts required to understand the secure, decentralized storage and distribution of digital content. It will provide a summary of Blockchains and Smart Contracts and the cryptographic principals underpinning them as well, a summary of the most prominent decentralized storage platforms and look briefly at regulatory concerns in this new field. We will conclude with an overview of the film industry and other platforms in development with some similar aspects to them.

2.1 Blockchain and Beyond

2.1.1 Overview

In the last year or two there has been a lot of hype around Blockchain and the possibilities it gives of disrupting various industries. There has been talk of completely changing the strong hold banks have on financial markets [74], changing existing models of governance [42] and even creating decentralized autonomous corporations [78] where the management is in the hands of algorithms, automatically hiring, firing and making business decisions in accordance to a mission statement.

But what is the Blockchain? A Blockchain is a decentralized database, which requires the consensus of the majority of the users to add to it. It thus contains all added records dating back to its creation, publicly visible to everyone. All users validate any new block being added to the decentralized database, voting on whether they agree with the data contained in the block [69]. It requires the majority of the users to be honest which in practice works well. The voting of all Blockchain participants removes the need for a central trusted party.

The Blockchain was a solution initially proposed by a person under the pseudonym of Satoshi Nakamoto to solve the problem of having to trust a third party to securely transfer a

digital token between two parties [65]. When people refer to the Blockchain they typically mean the Bitcoin Blockchain.

2.1.2 Cryptography

All things Blockchain are made possible by Cryptography, a field in Computer Science and Mathematics concerned with the secure transfer of information between two entities. Cryptography forms the foundation of how a system can work securely without regulation or a centralized, trusted party. This is a large field in Computer Science and I will only touch on it briefly here.

Given a randomly generated 256 bit private key (can be thought of as a password), we can generate a public key using Elliptic Curve Cryptography by repeatedly adding integer points on the Curve as shown in Figure 2.1. This ensures that retrieving the private key from the public key is computationally unfeasible (i.e. similar to solving the discrete logarithm problem).

Elliptic Curve Cryptography not only allows us to securely create public keys for Blockchain users but also allows us to create and validate digital signatures. It is more efficient in terms of storage and computational security per bit and was thus chosen over other standards such as the Rivest-Shamir-Adleman (RSA) Crypto-system [43].

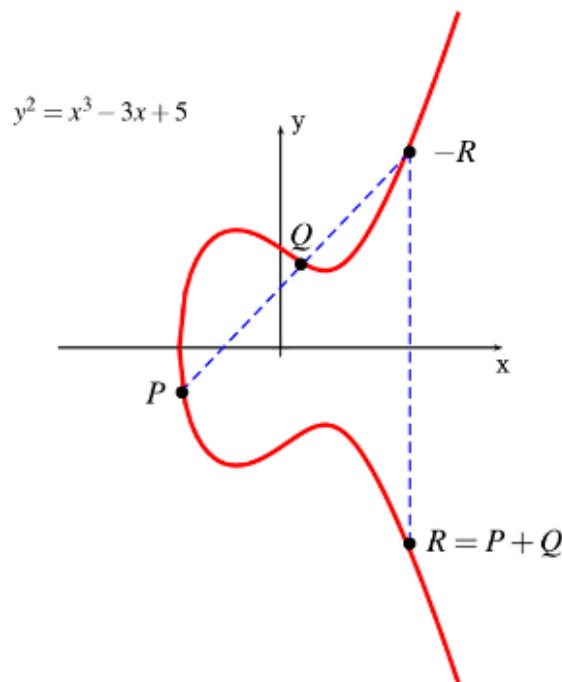


Fig. 2.1 Elliptic Curve used in Bitcoin: Adding points P and Q to yield R [34]

Hash functions capture the integrity of the argument passed to them. They are used to ensure a message has not been tampered with. For example: If Alice sends Bob message m with some random padding on the end s , and hash of message m , $h(m)$, Bob can ensure that the message m' he receives is the same as the one Alice sent by calculating $h(m')$ and ensuring it is equal to $h(m)$ as illustrated in Figure 2.2. Hash functions are used throughout Blockchain implementations to ensure various parts of the system have not been tampered with and to provide additional complexity to items such as addresses, by hashing the public key output of the Elliptic Curve to create an address [44].

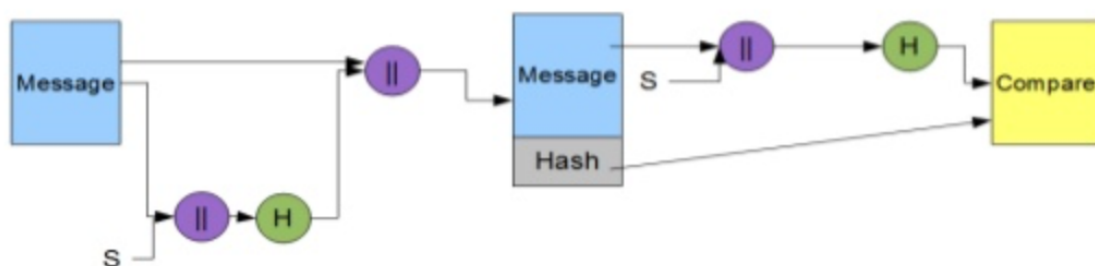


Fig. 2.2 Hash Function Example. Purple circles represent string concatenation and green circles are the hash function [8]

2.1.3 Bitcoin

Bitcoin is a digital cryptocurrency. It uses cryptographic techniques for security, validating transactions and generating and distributing new currency.

It is important to first understand how Bitcoin represents who owns the individual coins. Currency is not listed as the balances of the individual accounts, as one would be used to from the banking model. To calculate the balance of any person's account one has to trace back all transactions to the introduction of currency to the system as illustrated in figure 2.3 [65]. For example, if Alice wanted to send Bob 1 BTC, one would first have to check all the transactions that flow into Alice's account minus all the transactions that flow out. For each of the transactions flowing into Alice's account, it is necessary to check the transactions flowing into the sending accounts, minus those flowing out and so on all the way back to the point at which that currency was introduced into the system.

Once verified, the unspent transaction from Alice to Bob is broadcast to all users in the system at which point a competition is held as to which participant gets to insert it into the Blockchain. The Blockchain is a list of such transactions, grouped into blocks, held by each participant in the network.

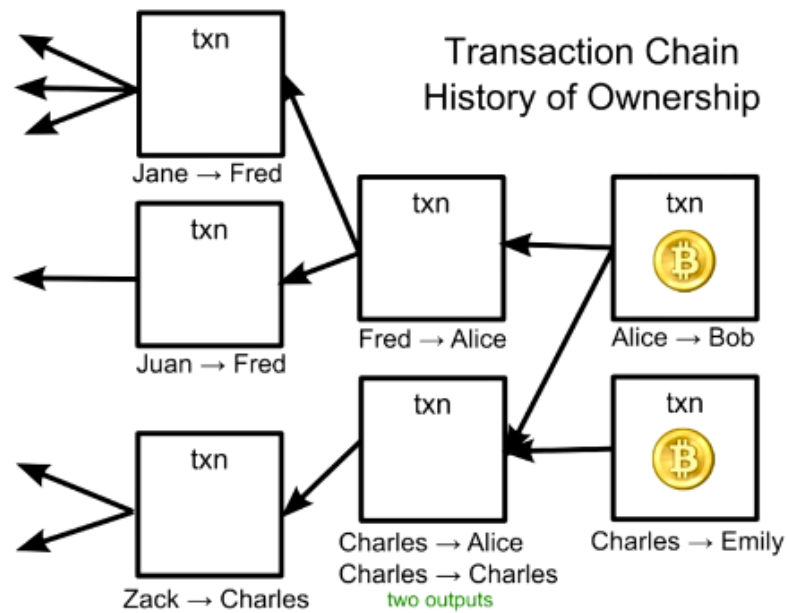


Fig. 2.3 Chain of Transactions in Bitcoin used to calculate account Balances [10]

Now the immediate question is: can everyone see everyone else's balance if they trace back the transactions? No. Instead of transfers happening between accounts, they happen between public keys. Public keys are an area of cryptography, where a private key (essentially a password) is encrypted by a non-invertible function yielding a public key. This step is computationally efficient but deriving a private key from a public key is not. This essentially requires guessing all possible passwords and generating public keys from them, something very improbable, even with all the combined computing power in the world [3]. This means users can exchange public keys generated from their private keys freely without giving away their identities. When they wish to make a transfer they digitally sign the transaction with their private key and the network knows it comes from them by checking it against their public key as illustrated in figure 2.4.

The next important step is how exactly the transaction between Alice and Bob is put into the Blockchain. All active nodes in the network contain a complete copy of the Blockchain i.e. the list of all previous transactions. When the transaction between Alice and Bob occurs they broadcast it to the network. Participating nodes, called miners, combine many transactions they have been informed about into a block and run it through a cryptographic function with a random guess until the output is smaller than some number, illustrated in figure 2.5. This takes a large amount of computing power, essentially creating a guessing game. All miners race against each other until there is a victor, who receives all fees associated with the

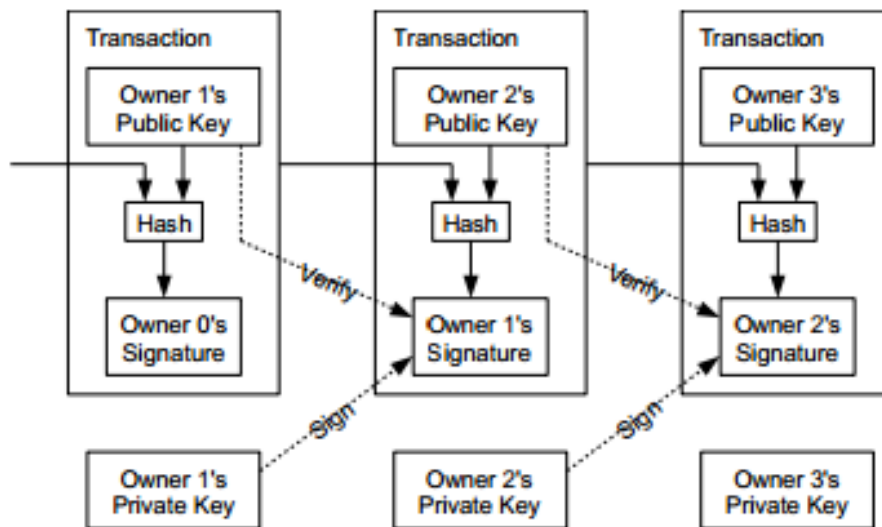


Fig. 2.4 Chain of Transactions being Signed and Verified [11]

transactions they put in the block, as well as some newly introduced currency by the system. The amount of newly introduced currency halves every so often, meaning that eventually it will stop and the fees are the sole remuneration for the miners. The block and solution to the guessing game is then broadcast to all other network participants, who can very quickly verify it as correct and then add it to their own Blockchain thus starting the process again [79]. The concept of repeatedly hashing a guess (the nonce) with the block until a victor is found is called the Proof of Work illustrated in figure 2.6.

The difficulty of this guessing game is adjusted based on the computational power of the network in an attempt to keep a steady flow of 1 block being introduced every 10 minutes. If an inconsistent block is broadcast, the network notices when validating it and ignores it, not further broadcasting it to other nodes.

So far we have not discussed the exact process by which a transaction is validated. The inputs and outputs to a transaction all have associated scripts, which are added when the transaction is created. When validating a transaction the scripts associated with the inputs must be run. There are two scripts: a locking and an unlocking one. The locking script is given to any transaction output and must be run in conjunction with an unlocking script when the transaction is used as an input and yields true to validate. If running the two scripts leads to any other value the transaction is deemed as invalid and not included in any blocks [12].

This scripting feature allows for complex forms of transactions to be created e.g. ones that require multiple signatures to transfer funds. However, by design, this language was not made Turing Complete. A system that is Turing Complete means, in theory, that any

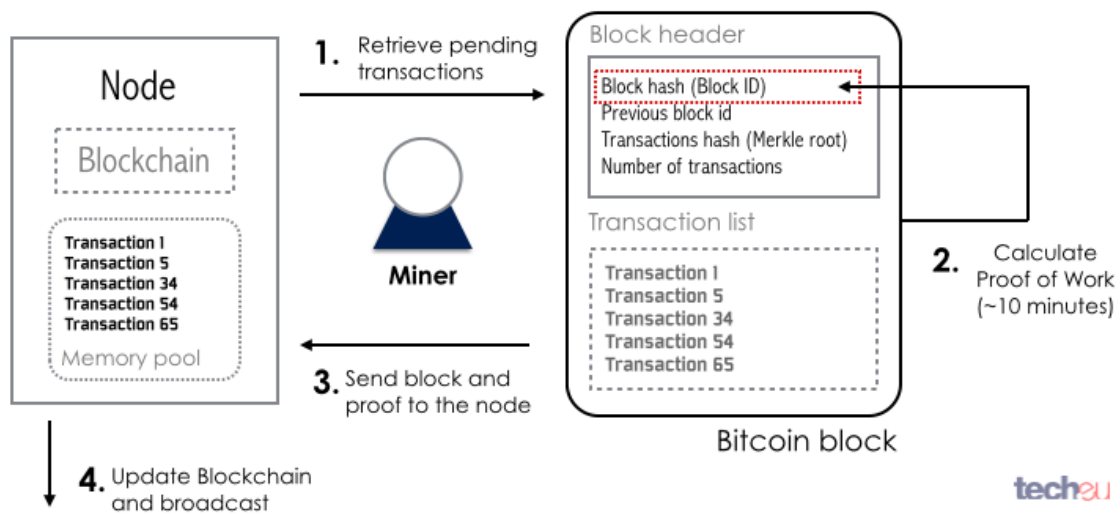


Fig. 2.5 Bitcoin Mining Process [9]

Proof of Work

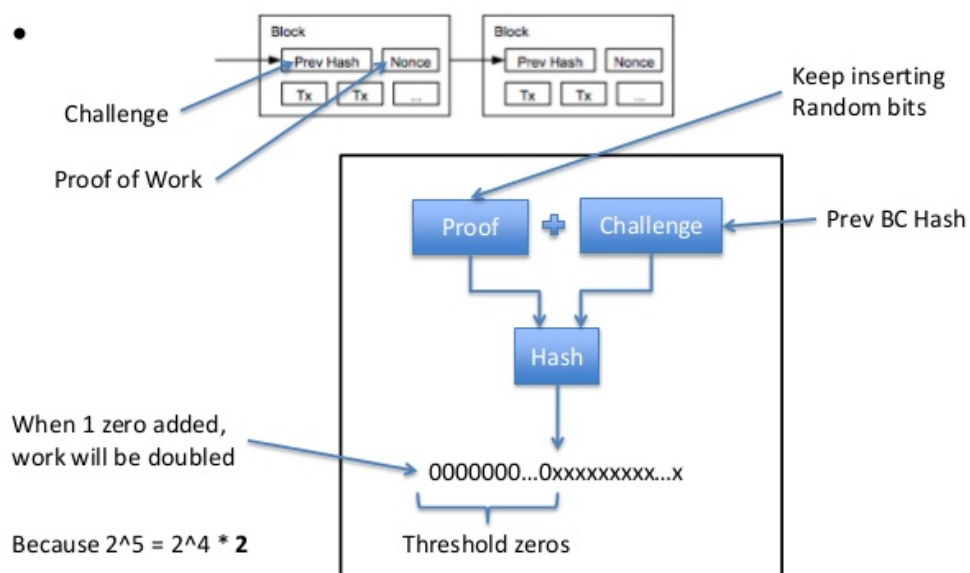


Fig. 2.6 Bitcoin Proof of Work [77]

computational problem can be solved but with no guaranties about time and space complexity [67]. Bitcoin does not want non-terminating scripts because this could crash the network by consuming too many resources, a flaw easily exploitable by hackers.

This is not a full summary of how Bitcoin works but it is enough to illustrate why it is of no use to this project. The limitation in the above-described scripts is why we cannot use Bitcoin itself. We require a richer programming language that is Turing Complete to define the complex update rules of distribution contracts. For this functionality we will examine systems designed to get around the limitations of Bitcoin, commonly called Bitcoin 2.0 applications.

2.2 Smart Contract

2.2.1 Overview

Smart Contracts date back to the 1990s where Nick Szabo, a prominent figure in the Cryptography world, first used the term. The idea is essentially the same as a normal program which uses predicates to represent contractual clauses, but they would control the underlying, real world asset. When certain conditions are met, the contract would automatically execute its clause [22]. The problem back then was the inability for programs to create payments/transfers of value, however all of this changed with Bitcoin. The following four companies all tried to address the shortcomings of Bitcoin scripts using the idea of Smart Contracts: Ethereum, Counterparty, RootStock and Codius.

2.2.2 Codius

The idea behind Codius is to host smart contracts on many different, centralized service providers and give users the ability to connect to multiple decentralized databases (e.g. the Bitcoin Blockchain) and other APIs [27]. The problem it attempts to solve is that of the oracle. In any decentralized network with scripts checking predicates, the information for the predicate typically needs to come from a central source (e.g. the price of a stock or the current weather in London) [17]. These sources must be trusted by the network, which is counter intuitive as decentralization is a means of minimizing trust.

By adding a layer between the distributed ledger (e.g. Bitcoin) and the client, as shown in figure 2.7, one can combine oracles and decentralized networks, without fundamentally weakening the security of the network. This can be thought of in the same way as the Application Logic Layer, which sits between the database and the User Interface in a typical Application.

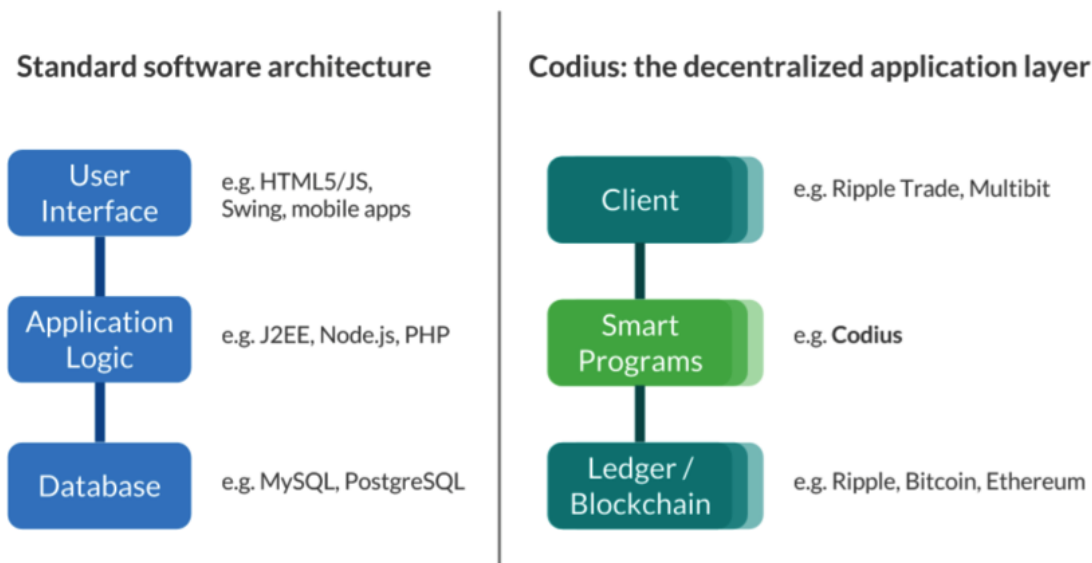


Fig. 2.7 Codius Architecture [26]

Codius was discontinued by Ripple Labs as of July 2015 due to a lack of demand. The creators themselves see it as an optimization, not necessary to build good decentralized systems. [64]. The system was never fully implemented and is now progressing very slowly in an open source manner. It is far too incomplete to be used in this project [41].

2.2.3 RootStock

RootStock's primary objective is to bring Smart Contracts to Sidechains.

Sidechains work by expanding on a mechanism called 2-way pegging. They work on top of the Bitcoin network by sending a certain amount of BTC to a script's address and receiving that same amount of a different currency, for instance, RootStock's Rootcoins (RTC). This functions in a very similar manner to the previously-described transaction between public keys, only now, the funds are not controlled by whoever has the private key for the address but rather by certain conditions in the script being met by a transaction. If a transaction were to prove to the script that it had destroyed the previously issued RTC, the original BTC would be released, thus illustrating the two-way behavior. When making the initial transaction to the script, one would publish the proof to the RootStock's Blockchain, which is how the new currency RTC is issued [19].

By using a side chain, RootStock's currency benefits from the security of the Bitcoin Blockchain, considered the most secure Blockchain and Protocol at the moment due to the

number of participants in the network. Another benefit is that BTC and RTC are always easily exchangeable in a pegged manner, meaning there is no speculative value to RTC i.e. it has the same value as Bitcoin.

The problem with Sidechains is that, as previously mentioned, after a fixed number of new blocks being introduced to the Blockchain, the newly-introduced currency issued to the successful miner halves and that loss must be made up for in the transaction fees. This may cause large increases in fees. Many miners will need to upgrade their systems to benefit from newer, faster processors to maintain their profit margins. RootStock provides miners the opportunity to merge mine, a method allowing them to mine on the RootStock Blockchain as well at the Bitcoin Blockchain at the same time with no marginal cost. This lets them maintain their profit margins for a few more years at least [58].

The RootStock virtual machine (RVM) will allow the execution of Smart Contracts in parallel across many on the network participants. The results of execution may be inter-contract communication, transactions or mutation of contract's persistent memory. It is designed to be compatible with the Ethereum virtual machine (EVM) on the OpCode level. This will allow Ethereum contracts to run flawlessly on the RVM using Rootcoin and thereby Bitcoin rather than Ethereum's currency Ether [66]. Due to the Turing completeness of the RVM it is a natural candidate for this project.

Currently, there is limited information available as to the exact implementation of RootStock but it is still early days considering the white paper has recently been finalized and released. There is no formal launch date available making it unfeasible for use in this project. However, due to the interoperability between the RVM and EVM, if we choose to use Ethereum for this project, it could easily be adapted to use RVM if it is found to perform in a superior manner or if the general population would prefer to deal in the current most prominent digital currency, Bitcoin, rather than Ether.

2.2.4 Counterparty

Counterparty was not originally designed to execute Smart Contracts. Unlike the payment network Bitcoin provides, Counterparty provides a full financial platform, allowing complex financial contracts, as found in financial markets. It does so by storing data in the Bitcoin Blockchain every so often in transactions. Counterparty cannot debit Bitcoin and thus requires its own currency XCP, for various financial interactions such as escrow and clearing house operations. The currency XCP was created and given value by burning Bitcoins i.e. sending them to a non-spendable address [29].

Counterparty ported Ethereum's language Serpent and the Ethereum Virtual Machine (EVM) over to their platform. This allows Smart Contracts to be run on the Counterparty

network of nodes. The system functions rather differently to Ethereum and thus has little interoperability beyond the language Serpent [30].

The problem with Counterparty is they are not looking to make use of Sidechains. They will still have their currency XCP, and will burn Bitcoin to create it. Transactions in their system will always be in XCP or a custom currency, either way not pegged to Bitcoin. A solution would have been to use the Bitcoin asset provided by the platform but regulatory concerns lead to it being discontinued [55].

It seems there is currently a debate in the community whether Sidechains or completely re-engineered Blockchains such as Ethereum are better, without a clear victor. The financial industry is also watching with interest, with UBS stating they are interested in both technologies and only time will tell which works out best [1].

From the project's perspective, I do not see any value in choosing Counterparty, a platform that has no intention of using Sidechains to ease up on the amount they are bulking up the Bitcoin Blockchain, essentially adding to the problems Bitcoin itself is facing. Counterparty is also not aiming to provide interoperability with other Blockchain implementations, but rather chose to port across the EVM, which already has a new version being designed to counter scalability concerns [21].

2.2.5 Ethereum

The above sections illustrate that the only viable alternative to Ethereum is Counterparty and with the issues raised above, as well as the promising RootStock virtual machine (RVM) being developed, I believe Ethereum to be the obvious choice. It gives this project the ability to adapt to the uncertain future of Blockchain technologies. If Sidechains are found to be the way forward, the RVM can be used, if not then the EVM can continue to be used. The rest of this section will give an in-detail technical description of how Ethereum works.

The idea behind Ethereum was to create a protocol to enable the easy and secure creation of decentralized applications, by giving users the ability to create Smart Contracts with a Turing complete programming language and thereby allowing for arbitrary rules pertaining to ownership and transactions [90].

Ethereum uses the same proof of work concept for mining its blocks as Bitcoin although, unlike Bitcoin, where the state is made up of transactions, the Ethereum state consists of accounts with 20-byte addresses. The transfer of information and value between these accounts constitutes state transitions. There are 2 types of accounts [90]:

1. Externally owned accounts: These accounts have associated private keys. Messages are sent from them by creating transactions, signed with the private key. These accounts have no code associated with them.
2. Contract accounts: These accounts can be thought of as autonomous agents residing on the Ethereum Blockchain waiting to be activated by transactions or messages. Once active, they run their associated code which can mutate their persistent state, send further messages to interact with other contracts, transfer value to external accounts or create further contracts.

Each account has the following 4 fields [90]:

- Nonce: Ensures transactions are only executed once.
- Ether Balance: The accounts Ether Balance. Ether is the crypto-fuel used to pay transaction fees and transfer value between accounts.
- Contract code: Blank for externally owned accounts.
- Storage: The persistent state of the account, empty by default.

Transactions are just messages sent from externally owned accounts, signed with the associated private key. They contain:

- A signature from the sender;
- An address for the recipient;
- An amount of Ether to be transferred;
- A STARTGAS value, determining the maximum complexity of computation allowed for the transaction. Various Op Codes in the Ethereum Virtual Machine (EVM) require different quantities of gas;
- A GASPRICE the fee the sender is willing to pay per unit of gas consumed by the computation;
- An optional data field: This is typically data, which will be stored in the contracts persistent storage or can be a mechanism for informing the contract of external information such as the price of a stock or the weather in London.

STARTGAS and GASPRICE exist to prevent denial of service attacks, making all accounts pay proportionally for the resources they consume. A denial of service attack involves flooding a network with traffic, slowing it down to the point that it crashes, leaving users with potentially unprocessed transaction and value being lost in the system [52]. Most operations cost one gas yet writing to storage costs more as that information has to be saved, consuming storage space. Every byte of transaction data costs 5 gas as this consumes network bandwidth [36].

Consider, for example, that Alice sent Bob 1 Ether for a box of chocolate. Bob receives many orders each day and knows to wait 10 minutes before dispatching the product. After 10 minutes Bob sends the product. However, in the meantime, a denial of service attack was launched and the system crashed. When the system is finally rebooted the transaction has not been process and is forever lost, leaving Alice with the money and the chocolate and Bob with nothing.

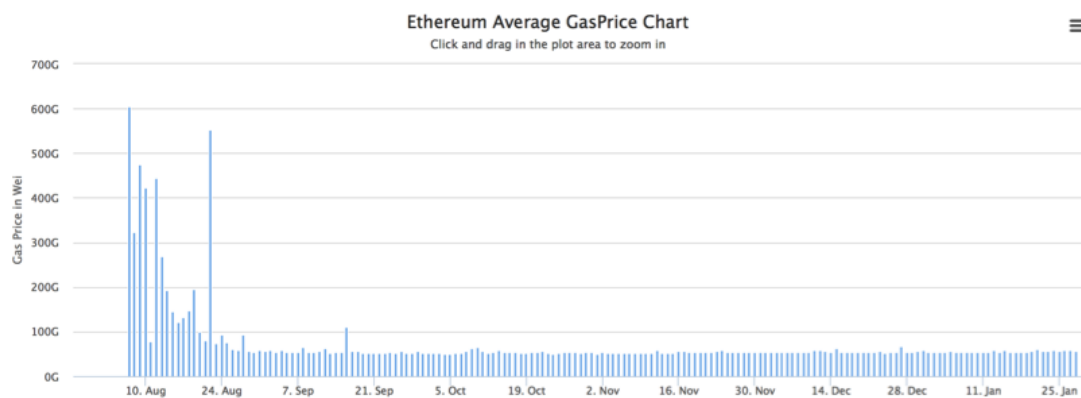


Fig. 2.8 Ethereum Gas Price [38]

The gas price fluctuates depending on availability of resources (supply) and the volume of activity on the network (demand) as illustrated in figure 2.8. 1 Ether is 1018 Wei.

Messages are used for inter-contract communication. They are just transactions sent between contracts with the same fields apart from the GASPRICE. This is because the only time a message is generated is when a transaction is sent to a contract. The contract at some point executes a CALL operation to another contract and uses the same GASPRICE specified in the transaction in its message. The STARTGAS of the transaction applies not only to the transaction itself but also all subsequent sub computations i.e. messages between contracts.

The Ethereum state transition function illustrated in figure 2.9, maps from the given state, s , and a transaction, tx , to the updated state, s' . It is defined as [90]:

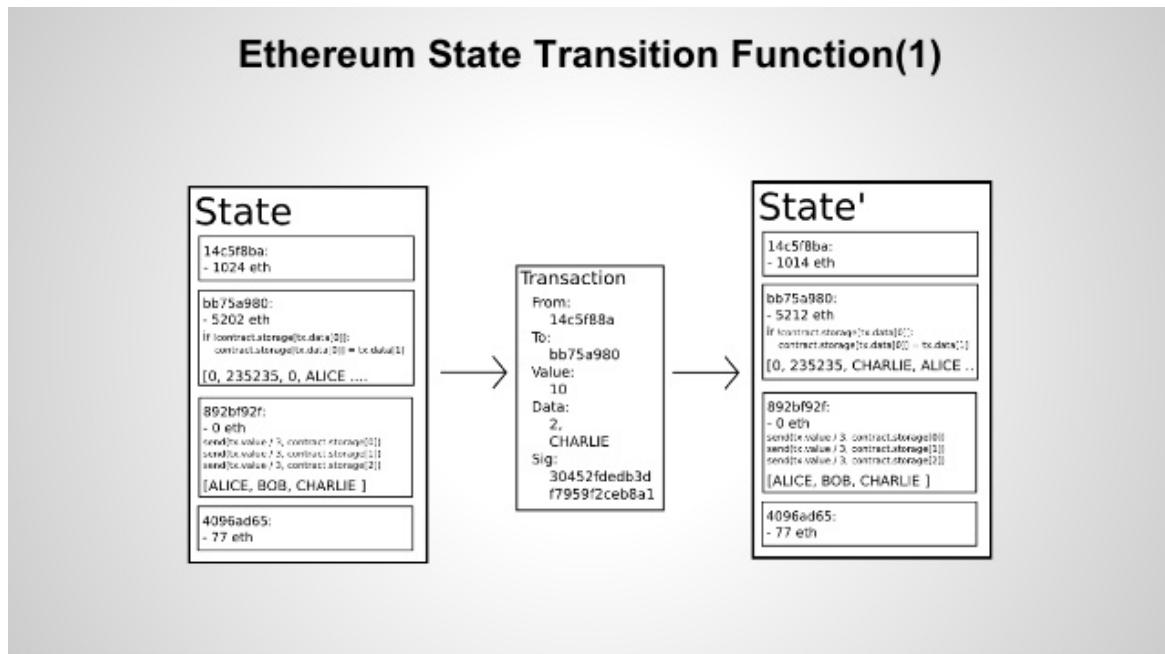


Fig. 2.9 Ethereum State Transition Function [51]

1. **Validate the transaction:** Check if it has all required fields, as described above, validate the signature and check if the nonce matches the senders nonce. Return error if invalid.
2. **Subtract Fee:** Use the signature to determine the sender, subtract $\text{STARTGAS} * \text{GASPRICE}$ from their account and increment their nonce. Return an error if the account has insufficient funds.
3. **Subtract bandwidth costs:** Let $\text{GAS} = \text{STARTGAS}$ and subtract the number of bytes in the transaction times by 5 (the price per byte).
4. **Transfer value:** Remove the amount of Ether to be transferred from the senders account and add it to the receiving account. If the receiver is a contract, execute the code until completion or GAS is 0. If the receiving account does not exist, create it.
5. **If failure:** Failure can occur either by the GAS not sufficing for the computation or the account not having sufficient funds for the value transfer. In this case revert state changes, except the consumed GAS and give it to the Miners.
6. **Allocate Fees:** All remaining GAS after completion of the transaction is returned to the sender and the consumed GAS is given to the Miners.

Messages work identically only the hierarchy needs to be taken into account. All sub executions generated by a message are reverted if it runs out of GAS but parent computations are unaffected as long as they do not run out of GAS themselves.

The Ethereum Virtual Machine (EVM) allows the creation of contracts using a low level, stack based byte code language. The execution is much like we are similar for existing computer model, where a program counter is infinitely incremented, and points to the operation to be executed. This process continues until an ERROR, STOP or RETURN operation is encountered.

Data is stored in one of three locations [90]:

1. Stack: The standard last in first out data structure, only ever allowing access to its top element through a POP operation or saving data to the top of it with a PUSH operation.
2. Memory: A sequence of addresses where data can be stored, theoretically infinite, but in practice limited by the capacity of the network.
3. Storage: Persistent key value storage.

Any execution context has access to the transaction or message data and block header data and may return an array of data. The state of the EVM can be summarized nicely by: (blockState (accounts, balances and storage), transaction, message, code, memory, stack, pc, gas) [51]. Any op code thus just mutates this structure.

In order to understand the Ethereum Block chain we first need to cover about Merkle Trees, Patricia Trees and the GHOST protocol.

A Merkle Tree is a type of binary tree. The leaf nodes contain the actual data to be stored. A hash is then generated for leaf and the result of two leaves are again hashed together forming the parent node as illustrated in the figure 2.10. This process is continued until eventually only one hash is remaining, the root, which is then stored in the block header. This ensures that none of the transactions in the tree can be altered without changing the root hash. It also allows miners to run a light client and just validate the block headers with a small section of the tree relevant to them, not the whole thing [6]. Patricia Trees are merely an optimization of Merkle Trees. Using a Patricia Tree does not require repeating all data in each block, but rather allows you to reference nodes in the Patricia tree of a previous block so that one only needs to record the new data and reference the data that did not change, as is illustrated in the next figure.

There are two main problems with Blockchains where security is significantly reduced and when confirmation times decrease, due to stale rates increasing. The first is that it takes a certain amount of time for any newly added block to be broadcast across the entire network.

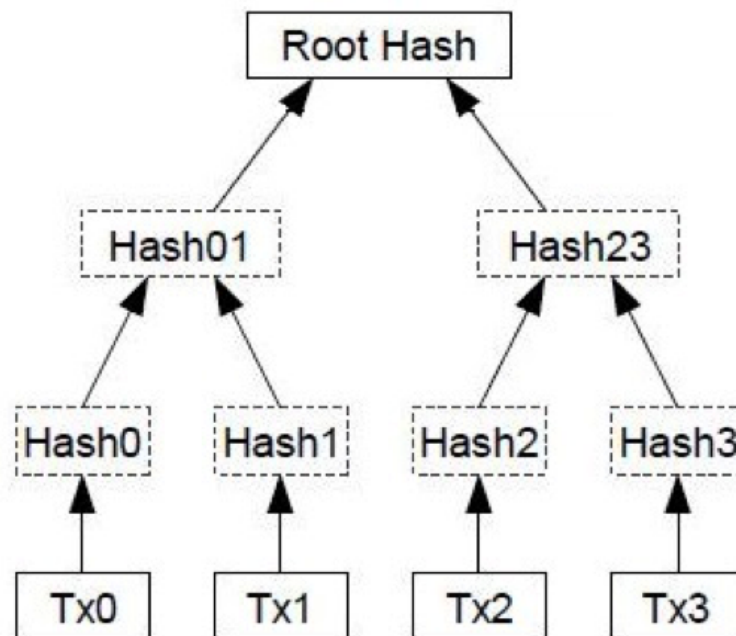


Fig. 2.10 Merkle Tree Illustration from Bitcoin [18]

In that time a different miner may mine another block, which will just be discarded as soon as they find out about the previously-mined block. Thus their computing power will not have contributed to the networks security in any way [73]. The second problem is a centralization issue illustrated nicely in the example taken from the Ethereum white paper: “if miner A is a mining pool with 30% hash power and B has 10% hash power, A will have a risk of producing a stale block 70% of the time ... whereas B will have a risk of producing a stale block 90% of the time. Thus, if the block interval is short enough for the stale rate to be high, A will be substantially more efficient simply by virtue of its size” [21].

The Greedy Heaviest Observed SubTree (GHOST) protocol was designed to counter the first problem mentioned above. The idea is to not only take into account the parent and further ancestors of a block when trying to calculate which chain is the longest, but also the stale descendants of a block’s ancestors [73]. Ethereum calls these stale blocks uncles and they are taken into account when the longest chain of the Proof of Work is calculated and then broadcast to the network as the agreed upon Blockchain [37].

To address the second issue, Ethereum went beyond the GHOST protocol and decided not to do unlimited GHOST but rather only go back 7 generations when performing the calculations. In addition, stale blocks receive 87.5% of their base rewards with the remaining 12.5% being attributed to its nephew but transaction fees are not given to uncles. This keeps

the calculations required simpler and having added the additional rewards for stale blocks, keeps the incentive for Miners to mine on the main chain, not any chain associated with an attack on the network [20].

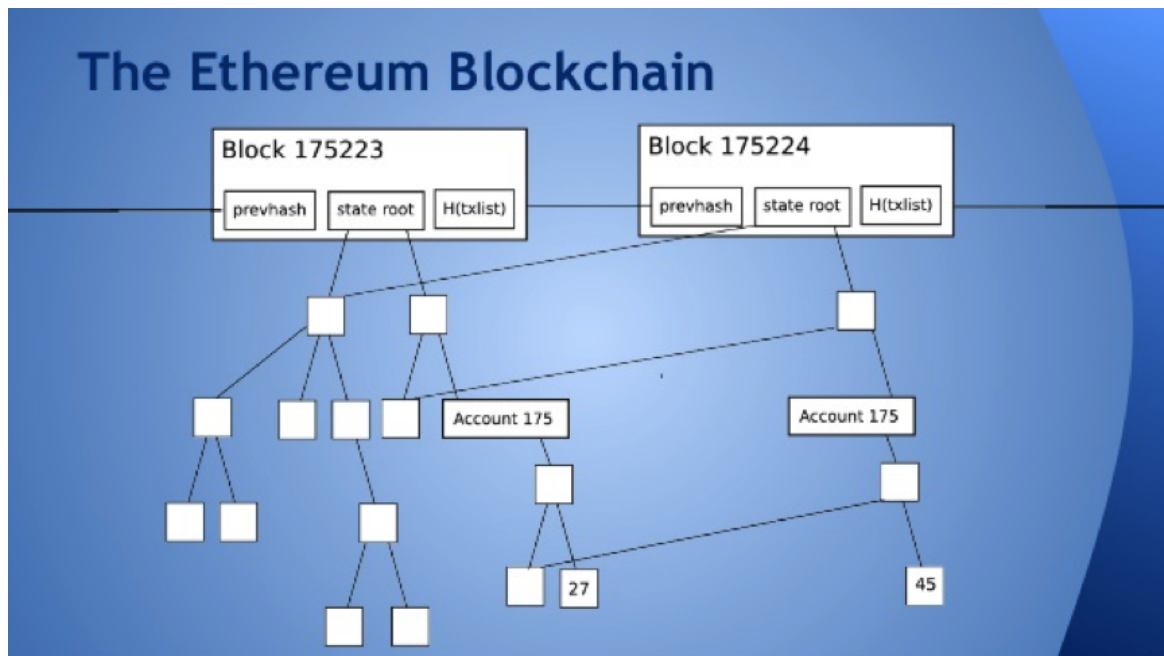


Fig. 2.11 Ethereum Blockchain showing shared state between blocks. [50]

Although the Ethereum Blockchain is similar to Bitcoin it does fundamentally represent its state in a different way. The state of the system is shared between blocks as illustrated in figure 2.11. Each block in Ethereum contains a copy of the:

- Most recent state (hashed root of Patricia Tree)
- Transaction list (hash of list)
- Block number
- Mining Difficulty for Proof of Work
- Link to previous block

The block validation algorithm is illustrated in figure 2.12 and summarized below [20]:

1. Check validity of previous block
2. Check time stamps: Current time must be greater than previous time and less than 15 minutes into the future.

3. Validate: block number, difficulty, transaction root, uncle root and gas limit.
4. Validate the proof of work on the block
5. Apply the state transition function: to the first transaction in the transaction list with the current state s , yielding s' . Then apply the function with the second transaction and s' , yielding s'' and so on until all transaction have been used. If any errors are returned or the GAS runs out return an error.
6. Pay fees: Allow the new state be the final state reached in 5 in addition to the block reward being given to the miner.
7. Validate the Merkle Tree root: check the Merkle tree root and the root provided in the block header are the same, else return invalid.

At first glance one may think this is very inefficient, as the entire state needs to be known in each block. However, the Patricia tree allows referencing the state in a previous block, essentially only requiring each new block to store the changes in state and a pointer to the previously-unchanged state, illustrated in the figure. Also due to the storage of the state in each block and not just the transactions, like Bitcoin, it is not necessary to store the entire Blockchain history. If used in Bitcoin, this could yield savings of 5-20x [20]. Since all nodes running a full client validate blocks, all network participants run the contracts' code.

Although I have not gone into detail concerning all Ethereum operation codes and gas prices, I believe this gives a good overview on how the platform functions. Contracts can be written in multiple languages, which are then compiled to EVM operations, the most advanced of which is Solidity, a language inspired by JavaScript, which we will use in this project. We will also be using an interactive development environment called Mix and the AlethZero client to deploy contracts to the test net. The Ethereum Mist browser can then be used to test the deployed application.

2.2.6 Discussion

We have covered the most prominent Blockchain implementations and weighed their relative strengths and weaknesses, however as this new field progresses and regulation is introduced, any of the current implementations may become unfeasible. Note I have not covered all platforms here, others exist such as Crypti, but they either build on those mentioned here, introduce nothing new from a theoretical perspective or have been abandoned and are left in the hands of the community.

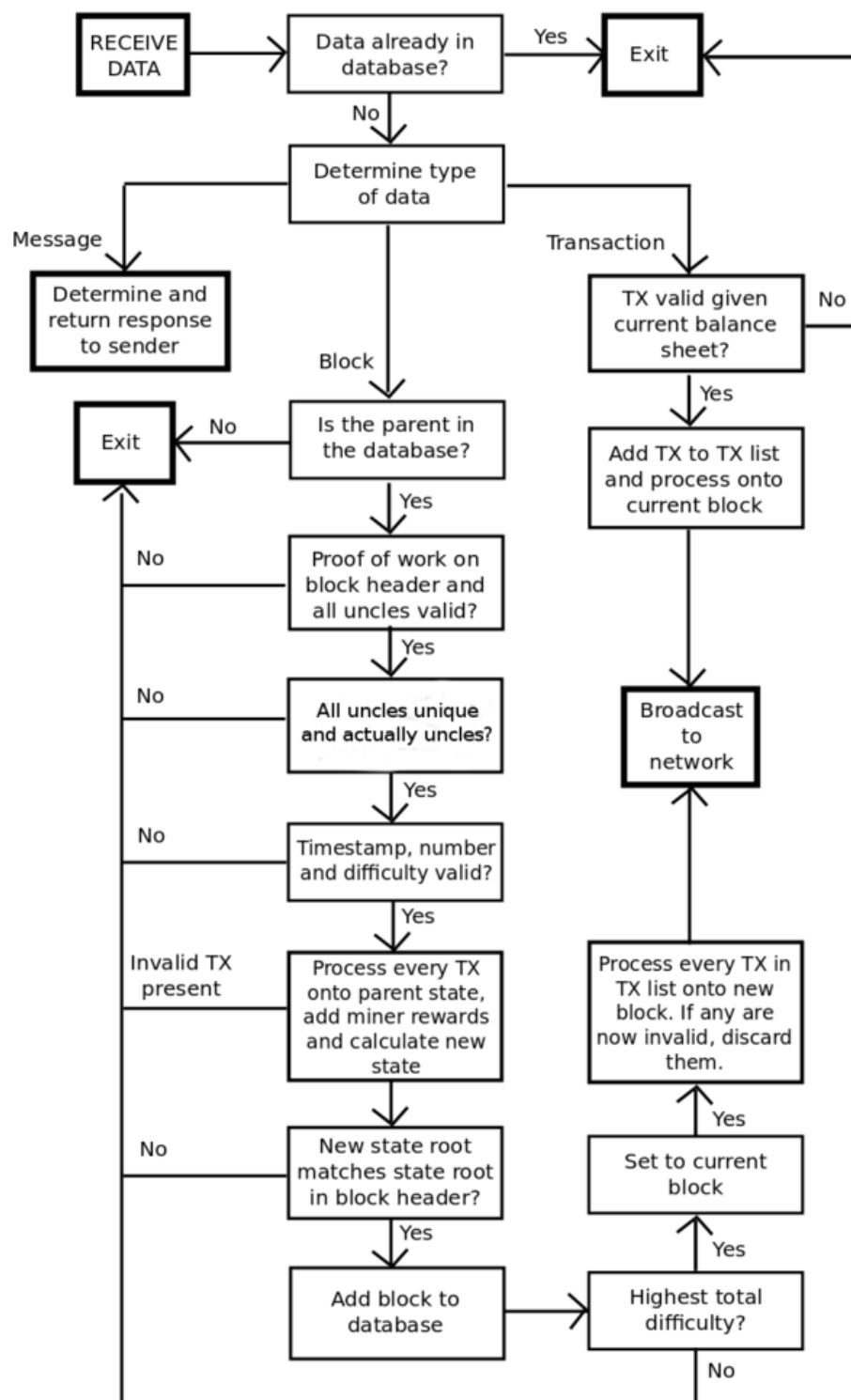


Fig. 2.12 Decision process for each Node in Ethereum Network upon receipt of data. [35]

It is worth mentioning that none of the current implementations are perfect as they aim to be general. In an ideal world we would create our own Blockchain implementation tailored to the specific needs of this project. To ensure security in such a network, one requires either a large number of random network participants, which takes time to amass, or internally controlled nodes, which need to be maintained. Developing a custom Blockchain is not possible as part of this project because we do not have a large team or significant resources (e.g. Ethereum took around 3 years until they launched their core features and raised \$20mil to enable it) [54].

2.3 Decentralized Storage

2.3.1 Overview

This section will summarize the most prominent decentralized storage systems and how they work. The discussion section will evaluate the systems and thereby determine which is best for this project.

Any digital content managed by my system needs to be securely stored. Smart Contract platforms like Ethereum are not aimed at providing large amounts of storage and therefore charge heavily for it. Current centralized or cloud storage systems are perfectly acceptable if one can trust the company hosting them. However, if a central system is hacked all information is available, a problem that does not exist in a decentralized system. Decentralized storage is currently cheaper than alternatives and gives an efficiency gain, as they make use of users' empty storage space on their devices.

To avoid having to trust any party, having all sensitive information compromised in a single hack and for price and efficiency gains we will be looking for a decentralized system to store the digital content in this project. We will also be looking for the ability to stream from such a system rather than having to fully download the content before viewing it.

2.3.2 Sia

Sia advertises itself as enterprise grade collaborative cloud for data storage. It uses its own currency, SiaCoin, to reward miners for providing Bandwidth and Storage and to charge clients for the use thereof. It uses smart contract to regulate the distribution of data stored across the decentralized system. Users only pay for what the resources they consume offering a much finer granularity of control, unlike traditional cloud systems where one rents a fixed amount, say 20GB per month. Users' data is secured through industry grade encryption on

their automated peer-to-peer market. Sia is a Sidechain and a variant of the Bitcoin protocol [86].

Sia is still in Beta and is due for release in June 2016. It will have Amazon S3 API compatibility by October 2016 [72]. At this point there is no API that could be used in this project to integrate with Sia. Their only partnership is with Crypti, an application no longer being developed but rather left in the hands of the community, which I find to be a little disconcerting.

2.3.3 Maidsafe

Maidsafe is creating the SAFE (Secure Access for Everyone) Network. Their focus is to make the transfer of data across the internet completely secure for all applications. They do this by offering a peer to peer service where individual users offer their spare storage and bandwidth in return for SafeCoin. This is also the currency for which anyone using the resources in the network pay. The data stored on the network is encrypted before sent to the network and only the encrypting party can ever decrypt it. They use a concept called Proof of Resource illustrated in figure 2.13 to ensure data is being stored correctly and determine how much to compensate the Farmers offering their resources [61].

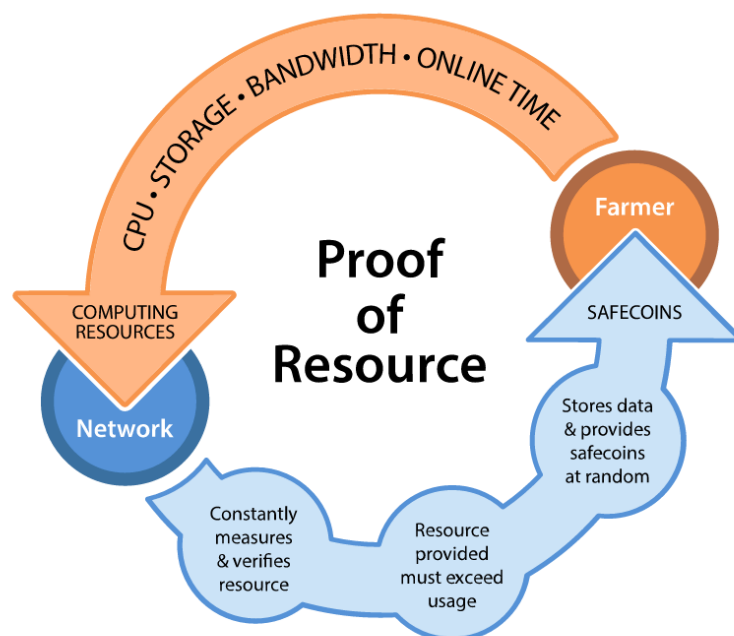


Fig. 2.13 Maidsafe's Proof of Resource. [62]

Maidsafe does not use the Hyper Text Transfer Protocol (HTTP), but rather their own SAFE protocol. Further down the network stack they use the User Datagram Protocol (UDP) with a Connected Reliable UDP eXchange (CRUX) rather than the Transmission Control Protocol (TCP). This means they are able to provide the reliability and congestion control of TCP without having any of the downsides, most importantly avoiding the problem of Network Access Translation (NAT) traversal [63].

Maidsafe is by far the most ambitious system mentioned in this section however they have not yet launched. I had a call with Nick Lambert the Chief Operating Officer at Maidsafe and it seems they still have a long way to go. He was unsure of what the best exchange was to use to trade Safecoin. He also suggested it would be a while before Decentralized Applications (DApps) can be built on their platform and that at this stage streaming video from the network was not possible. In an ideal world I would build this project on their platform but due to their current state of development and how long they have been building the platform and stating they are nearly there, I do not see that as a possibility for at least a year.

2.3.4 Storj

Storj aims to create a decentralized network for storing data. Users contribute their free storage and bandwidth to the network and in return are compensated with StorjCoin. This is the currency which people wishing to use the resources in the network must use. Security of the data is ensured by encrypting data client side before broadcasting it to the network. It started as an open source project and is now a company around a collection of decentralized applications including MetaDisk and DriveShare. They aim to provide a storage facility that cannot be censored, monitored and has no down time. It will be 10 times cheaper than traditional cloud storage systems and much faster too [75].

The system makes use of the Florincoin Blockchain. Florincoin is based on the source code from Bitcoin and Litecoin but they have made various improvements, most notably a 40 second block confirmation time [40]. Storj does have plans to move from this system to one more closely resembling Bitcoin or even Ethereum depending on how the industry develops in the future. Storj uses a Proof of Storage system to ensure all data is still available on the network. This works as follows and is illustrated in figure 2.14 [89]:

1. The client generates random seeds which each deterministically determines a series of seeds that can be added to the file to be stored and then hashed (Storj calls this process the heartbeat).

2. The client splits up the data and generates a Merkle tree, with the data blocks as the leaves inserting the root into the Florincoin Blockchain.
3. The client then sends a farmer the data and the Merkle Tree but not the Seeds.
4. Every so often the client can send the miner one of the root seeds and verify that the farmer responds with the correct hashed value.

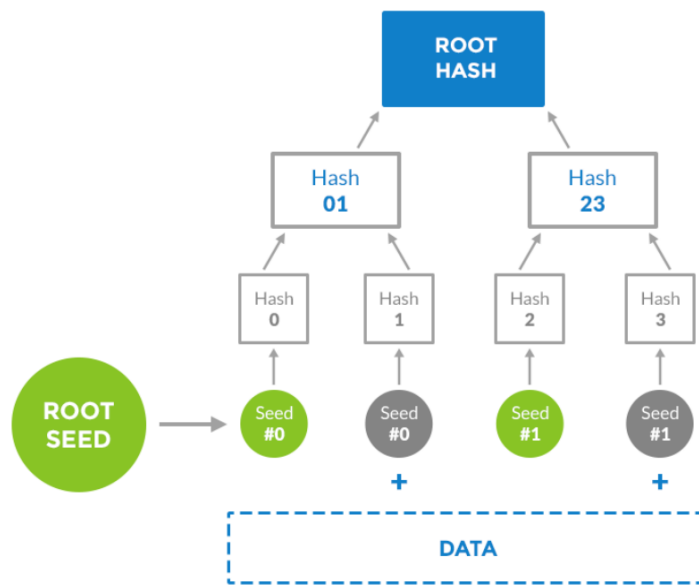


Fig. 2.14 Storj Proof of Storage: Generating a Merkle Root for Some Data [89]

Storj is still in beta but they prioritized their API making this a viable platform for this project. I have also found an example of streaming a video from the Storj network which is very promising [84]. It is still early days and there are concerns, mainly that Storj has not yet settled on its underlying architecture, however at this point in time they do provide the functionality we require unlike the alternatives.

2.3.5 Inter Planetary File System (IPFS)

IPFS has a slightly different model to the others. It is also a peer-to-peer network however there are no miners/farmers offering their storage space in return for some cryptocurrency [7].

To upload data to IPFS one has to be running a node locally. It is then as simple as pushing to a git repository and it can be mounted as a virtual file system using MUSE. The

hash of the uploaded item becomes the address to access it. Clearly this has advantages as duplicates in the network all have the same hash and so can automatically be served from more peers when someone is trying to access it. If you wish to guarantee the persistence of the file you have to pin it to your local node [7].

When someone attempts to retrieve a file, their node contacts its peers to resolve the hash. Eventually a node containing the file with the hash is found and downloaded by the seeking node. Then both nodes become seeders offering the file to any other peer looking for it.

This model could work well for our purposes. The more people access a certain asset the more copies of it there are around the network and the faster others can access it. The downside is all assets will also have to be stored and pinned on our local node so I see this as a half way step between centralized private storage, and complete decentralized storage as is offered with Sia, Maidsafe and Storj. Another downside is how we might limit access to the asset as anyone with the hash can access an asset any number of times.

2.3.6 Discussion

Most of these solutions adopt a similar model. They have some coin which is used to compensate users for providing storage and bandwidth to the network and charge users for these resources. The points of difference are the protocols governing how the data is distributed throughout the network and the various integrity checks thereof. IPFS adopt a different model where you host your own files but others can download and host them too. Therefore you do not have to pay to be part of the network.

Since Maidsafe and Sia are still in Beta with very limited features it is by means of exclusion that we will use either Storj or IPFS for this project. However, it should be noted that Maidsafe is an extremely ambitious project worth monitoring for future use.

2.4 Regulatory Concerns

The rise in the value of Bitcoin and the many alternatives that have been created since has led to a lot of interest in the field from the general public as illustrated in figure 2.15. So far very little regulation has targeted the Blockchain. It is early days and by no means easy for a non technical person to understand how many of these decentralized systems work.

Below are some of the recent developments:

- "In Feb 2016, the European Parliament saw a motion issued by its Committee on Economic and Monetary Affairs (ECON) that summarized the risks and opportunities with virtual currencies and distributed ledger technology (DLT)" [32]. For the week



Fig. 2.15 Bitcoin Market Cap vs Price. [28]

commencing April 18, the European Parliament is hosting a crash course in an attempt to educate the members on the topic [32].

- On the 26 April the Imperial College Center For Cryptocurrency Research and Engineering is presenting to some Government officials to illustrate the benefits of the Blockchain to the UK Economy.
- There are a few exchanges that are backed by regulation, the most notable of which in the US and UK is Coinbase [81].
- The World Economic Forum's event in Davos in 2016 featured many talks, in an attempt to educate business leaders on the topic of Decentralized Ledger Technology [70].

There are still many concerns about illegal use of unregulated currencies as in the last years the main uses for cryptocurrencies, have been on platforms such as SilkRoad, a decentralized market place which launched in 2011 and was finally shut down by the FBI in 2013 [4]. The platform enabled the trade of anything without being able to trace the supplier or customer. It thus mainly sold Drugs and Illegal documents but was even used for hiring a hit man.

It seems legislators are currently in the research phase and it will be very important to see the results of this once decentralized systems are better understood. I think one of the main issues that is weighing on legislators' minds is: how do we benefit from such systems without enabling criminal activities which are incredibly hard to trace? Only time will tell how this will be solved, however these technologies seem to be too popular a tool to be completely shut down.

2.5 The UK Independent Film Industry

2.5.1 Overview

In this section I introduce the classical distribution process in the UK's independent film industry. I will then go on to discuss problems currently being faced in the industry relating to security. This section is concluded with an insight into the market from a businesses perspective and finally a brief evaluation.

It is important to first understand what is meant by the independent film industry. An independent film (also called indie film) is a film that is not produced by any major studio. The distribution thereof is also done by entertainment agency with no direct link to any studio. This structure means it is harder to gain investment and exposure, but gives the creators more freedom, generally resulting in more creative and innovative content.

2.5.2 Distribution Process

The process starts when a film is completed. The producer typically secures the services of a sales agent, who takes the films to various film festivals and contacts distributors [15]. Sometimes the filmmakers themselves do this work. If successful the film then has a:

- Theatrical and non-theatrical release: within 6-12 months of completion
- DVD rental release: about 6 months after theatrical release
- Pay television release: another 6 months after video rental release
- Free TV release: up to a year after pay TV release

Many independent films make their sales after theatrical release. The theatrical release establishes the profile and reputation for the film, and is in a way an expensive marketing exercise. This leads to an ambivalent relationship between the distributors and the cinemas. From the distributors point of view they would prefer the cinema to release the film much sooner so that they can take advantage of the buzz around a film. As a result the above-mentioned windows for exploitation are collapsing, particularly for independent films.

Each of these stages is addressed in the distribution agreements. The terms of most deals are standard but there does exist a degree of negotiability:

- Advances: Sales agents or distributors often offer an advance to the producer as well as certain agreed costs, which will be paid back before the producer receives any percentage of sales. These are typically negotiated on a film-by-film basis [15]

- **Percentage of Sales:** Sales agents typically take 30% of advances and sales revenue. Distributors take about a 50/50 cut for theatrical/ non-theatrical releases, 20/80 cut in their favor for video releases and a 30/70 cut in the producers' favor for TV sales. These splits do not fluctuate very much [15]

The process is clearly costly to the film makers, leaving them with around 20%-30% of sales after all advances and agreed costs have been paid off. This figure decreases further if cinemas take a cut. The flow of payments can get quite complicated and it can be hard for a filmmaker to know exactly how their film is performing at any point in time.

2.5.3 Security

A large concern is the security of the film as it is screened and distributed. Piracy has had a large effect on the film industry and online streaming often leading to lower sales figures if leaked too early [59]. The film industry has responded to successful hacking attempts by canceling many films in production, costing them millions, the most recent of which was the Sony hack [68]. The cost of piracy to the UK creative industry is estimated to be around £500m per year.

2.5.4 Film Industry

The global market for film and video entertainment online is large and is growing quickly. By 2017, Cisco estimate 69% of consumer web traffic will be video [82]. This is illustrated in figure 2.16. Informa tells us that the global online video market will be worth £24bn by 2017 [71]. If we look at the film industry alone, the US International Trade Administration estimates that we will see a decline in the growth of box office sales by 2018 whilst online delivery of films will grow exponentially by some 21.3%. It concludes the number 1 trend in media and entertainment is for new technologies to “reshape the economics, production, distribution and marketing” [53].

Whilst the numbers are huge it is, nevertheless, notoriously difficult for film and video creators to exploit their creations online, particularly small and medium-sized organizations. This is partly due to piracy, but also due to huge marketing costs involved in promoting film and video. Without the upfront investment in marketing of film entertainment, most films are a loss-making business (the average advertising spend for a studio backed film in the UK in 2014 was £1.4m). Such costs are so prohibitive that only the studios can afford them. So despite the predicted market of £24bn, 80% of it will be retained by traditional producers and distributors although they account for just 5% of the content. In film six US major motion picture studios dominate distribution in the UK, accounting for 96% of the market. [53]

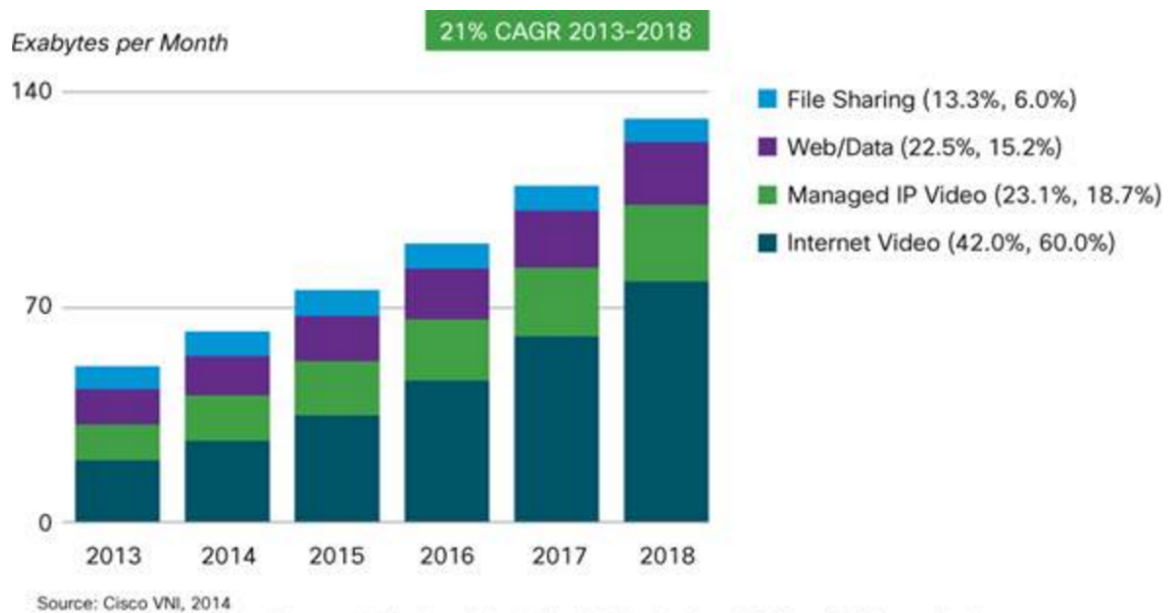


Fig. 2.16 Internet Traffic. Percentages in parenthesis denote relative traffic shares in 2013, 2018 respectively. [23]

2.5.5 Discussion

The upshot of all this is that the classic value chain, the windowing system for exploitation, is becoming more and more strained. Cinemas are under pressure because of lack of sales (competition from other forms of entertainment) and from further down the system where distributors and VOD services want to exploit films sooner, in the short window in which they are very popular. Also there are new entrants (Netflix, Amazon Prime, Hulu) who are not only distributing film content but also developing their own IP. The recent film, *Beast of the Nation*, was funded by Netflix, had a limited release in cinemas, doing little more than qualify it for the great cinema awards such as the BAFTAs and Oscars, and, simultaneously, was made available on the Netflix platform. In this particular scenario the distributors were circumvented altogether [57].

The current system is clearly under pressure from both ends of the value chain. This situation is changing the nature of the market. As time goes by fewer and fewer original, independent films are being made.

This has cultural as well as commercial implications, as I previously mentioned, the most creative and awarded content is that of the independent film. Also it should be noted that the emerging, mainstream online distribution alternatives, do not necessarily re-dress the balance. Being on their platforms does not equate to reaching an audience; as a film owner you do not have access to the analytics and are therefore provided with no insight into your market,

as well as having to pay for access to the platform itself. It is the high-end, blockbuster movies and TV that are driving the success of Netflix and alternatives, not independent movie making.

The process depicted here is primarily that of the independent film. The other main process for film creation is that of the Studio. Studios have control of the vertical and so their process is much more integrated (and insulated). They will often create the script, fund the film and distribute it all in house or with businesses they have formed tight agreements with. It would be much harder to gain traction with this project under that model, hence the choice to initially target independent film.

Film sales, coming from theatrical releases, seem to be staying constant year on year. DVD rental and retail as well as TV are all showing significant decreases in sales, with digital video showing the largest increase of all over the last few years as illustrated in figure 2.17. Essentially it seems apart from in theatrical releases, films are mainly being viewed online through video on demand services. These are the sectors of the market our solution will engage in.

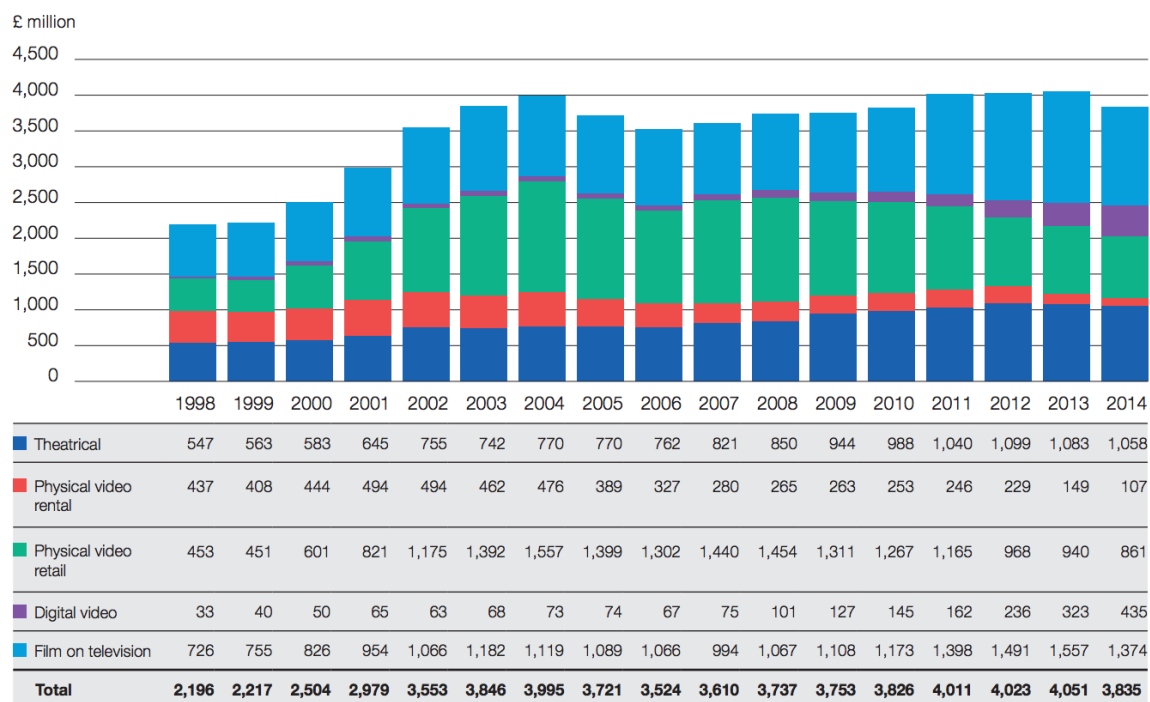


Fig. 2.17 Gross film Revenue on all Platforms. [16]

2.6 Related Projects

2.6.1 Overview

In this section I will discuss the applications in existence or development that are trying to do similar things to some of this projects objectives and point out the overlap in the concepts.

2.6.2 Bittunes

Bittunes aim to create a platform where musicians can connect directly to their fans, utilizing Bitcoin as the underlying currency. They have chosen to exclude all record labels from the platform. They have an android app which rewards distributors of the music content, which suggests they have a peer-to-peer network. The only part of their platform that makes use of Bitcoin is the payments [14]. Their next steps for the platform is "simple rights and transaction history Blockchain" which I take to mean they will be inserting some information on the individual songs into the Bitcoin Blockchain similar to MediaChain [13].

The only idea this project has in common with Bittunes is using a cryptocurrency to compensate individual artists, enabling the efficient transfer of value by cutting out middle men and reducing transaction costs.

2.6.3 FilmFund.io

Filmfund works by allowing film makers to make presales using digital tokens, essentially crowd funding the film [39]. When they reach a certain target, film production begins, without any debt obligations. Once completed, the film is distributed through the Lovegood platform (parent entity). It will already having fans who own TIX, the currency that is equivalent to 1 view of the film and lists the fans name in the credit of the film. This can then be exchanges to watch the film or traded on the exchange [60].

The only similarity between this project and Filmfund is the idea of distributing a film based on a predetermined agreement. Filmfund has static conditions whilst we are trying to create generalized contractual agreements for distribution which are not only dynamic but also hierarchical to express an entire distribution chain and complex relationships between Entities.

2.6.4 UjoMusic

Ujomusic allows music creators to upload their content and list the policies associated, including the revenue split, the lyrics and additional information regarding attribution of the

individual components. It is built on Ethereum, meaning transactions are made in Ether and a permanent records of the transactions are inserted in to the Ethereum Blockchain. The next steps in development for them are: removing the reliance on Ether, allowing users to pay in fiat currencies by assumably converting it to Ether by hitting an API of some exchange [83].

Ujomusic is the most similar to what we are trying to do. Our project will provide the ability to split the revenue generated from a film, however instead of having a simple percentage split we will also provide more complicated options essentially mimicking the ownership structure of companies by having Entities owning each other as well as individuals. It will be interesting to see how they deal with the exchange of fiat currencies to Ether, simplifying the process for users as this is a current concern our partner, the film network, has.

2.6.5 Digital Catapult

I met with Sam Davies at Digital Catapult to discuss their projects. They are working on a system using the Open Digital Rights Language to express contributions by games developers. The system will then split the rights of various parts of a game between the developers. This will be predetermined and I am not sure to what extent they are dealing with revenue flows it at all.

Details are scarce at this point but from what I can tell, our project is more focused on the distribution chain and revenue flows than the project being created at the Digital Catapult.

2.6.6 Mediachain

Mediachain is a meta data protocol which allows user to make statements about creative works, so for example who created them or who has the right to distribute them. These statements are then signed by the user, entered into the Bitcoin Blockchain to time stamp them and the statements themselves are stored on IPFS [56].

It seems that Mediachain have a similar idea to us in that they are trying to aid creative works in a digital environment. They are trying to ensure there is a record of who owns what digital assets along with other information associated with the item. This is a byproduct of our system but by no means the point of it, however we will attempt to get in contact with Mediachain to discuss potential collaboration as this project could work very well as a means of selling the digital asset. Potentially we could record where to buy a digital asset (i.e. a link to our system) in their statements.

2.6.7 Discussion

This section illustrates that there are a lot of companies attempting to solve some aspects of this project in their specific industry. If we are able to create a generalized solution I am confident it could be complementary to many of these companies and have applications in further industries still. It has become clear that there does not currently exist a system in development or production that meets our objectives.

Chapter 3

System Design

3.1 Overview

In this chapter we investigate the various approaches to securely storing and distributing digital assets and managing the associated permissions and payment structures. We analyze each different way of addressing the arising problems by looking at the benefits and limitations, in the context of the background research from section 2, and finally propose our solution with all details.

3.2 Smart Distribution Chain

As we have seen in the background research in section 2.6, current implementations are not looking at enabling distribution permissions for assets but rather on the problem of attribution.

Mediachain is trying to find a way to enable users to register, track and identify digital assets online. Digital Catapult is using general and extensible data reference models to express copyright and other legal aspects of video games. Whilst our solution will enable some of this functionality as a byproduct, it is not our focus. We are trying to address the distribution chain of the assets themselves.

For example, Imagine being able to upload a film, specify that you are happy for others to sell on your behalf for a year as long as you receive 20% of the profits and, if they do not make their sales targets, automatically accept new bids from other distributors for the contract. Perhaps you would even envisage allowing people distributing your film to approach other distributors on your behalf and create agreements, much like a sales agent would in today's model.

Consider the most disruptive scenario: The world would no longer have big film studios. When a new film is released, people with large social media followings and other small specialist sites would negotiate agreements with you and start distributing the film, allowing them to make money off the audience they can provide you. Then members of the audience you reach may decide to sub-distribute, by negotiating a deal with the social media stars, again a small piece of which the film maker would receive and so on as illustrated in figure 3.1 of viral spreading. This could all happen within a few minutes of releasing a film.

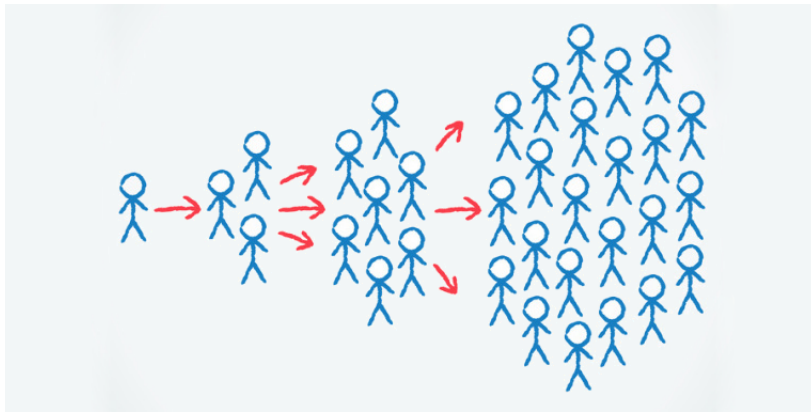


Fig. 3.1 Viral digital content. [33]

3.2.1 Creating Distribution Contracts

This section relates to the distribution contracts themselves. In an ideal scenario we would have some means of specifying the details of any distribution contract for any type of asset. This is a hard problem as it not only requires in depth knowledge of distribution throughout all industries relating to digital assets, but given that knowledge it is difficult to create a system that is general enough to allow all variations without being too complex. There exist frameworks like the Rights Reference Model provided by the Linked Content Coalition or similar open source projects, each of which have their own sets of limitations in expressibility due to generality but and are really quite complex to work with, thus we have focused on the film industry only for now.

Together with The Film Network and BAFTA we were able to come up with a list of contractual terms that can be enforced well and cover most of what film makers look for today. We have also been in contact with Wiggin LLP, a law firm specializing in media, technology and Brands/IP.

The list includes:

- Name: The title of the asset being distributed.



Fig. 3.2 Logos from left to right: The Film Network, BAFTA, Wiggin LLP. [5] [80] [87]

- Owner: The owner of the contract i.e. the entity giving the permission to distribute the asset.
- Distributor: The distributor i.e. the entity receiving the right to distribute the asset. Note: The owner and the Distributor do not have to be individuals, they can be a collection of entities (e.g. a company with shareholders).
- Price: The minimum value the owner of the digital asset is willing to accept for the purchase of one unit.
- Split: The split between the Owner and the Distributor.
- Asset ID: A number which uniquely identifies the asset.
- Contract ID: A number which uniquely identifies the distribution contract.
- Duration: A time period for which the distribution contract is valid once signed. Can be infinite.
- Number of Units: A maximum number of units which may be distributed using the contract. Can be infinite.
- Sub-distribute: A yes or no field which will allow owners of distribution contracts to give other distributors a 'copy' of the same contract.
- Sub-distributors: A list of Contracts IDs corresponding to sub-distribution contracts.
- Parent: The ID of the parent distribution contract if present. Note: We are building a tree structure out of the distribution chain.
- Signatures: A digital signature from the owner and the distributor.

In this context we are mainly interested in contractual terms that can be digitally enforced. The suggestion was to include the option to restrict distribution to a geographical location as well. Looking at the geographical location of the end user requires dealing with complications arising from virtual private network (VPN) usage and applications such as the TOR project.

This being said, The Blockchain provides an immutable record of all transactions meaning that even if the application itself cannot accurately determine the geographical origin of a request, there is a record of the transaction that can be used in legal cases in extreme situations. Because of this we decided to add geographical information which is displayed but not enforced by the Smart Contracting system itself.

There exist no current implementations of any form of distribution chain for regulating access to an asset apart from the trivial case previously mentioned of connecting the creator directly to the customer, and thus we have little to compare it to. We have spoken to the appropriate industry partners in an attempt to enable the current 'offline' ways of approaching distribution and will be keeping an eye on competitors as this field evolves.

3.2.2 Coming to an Agreement

Once a contract has been created by an Entity with their initial offer for each of the above mentioned fields it comes down to negotiation. It is rare for an initial offer to be accepted as is, so we will need a means of communication between the Entities. We have multiple options:

1. Take it or leave it: the asset owner could specify a standard distribution contract and multiple distributors could choose to accept it or not.
2. Individual Negotiation: The owner and an individual distributor could keep editing their contract until they come to an agreement or fail to reach a deal.
3. Auction: The owner could allow distributors to put in bids finally selecting one or many.

The first case is clearly the simplest (figure 3.3), however it does not accurately reflect how any asset is distributed. The owner is looking to sell as many units of the asset as possible and is willing to e.g. cut the price if a distributor can guarantee a lot of sales.

The second case is the one we have chosen for the film industry (figure 3.4). We have been advised that this is currently how independent films are distributed. Film makers or sales agents meet distributors at film festivals and negotiate deals with them on an individual basis. This gives sufficient flexibility for the distributors based on uncertainty about each individual film's appeal with its target audience. This also allowed the film maker to ensure they have a fair agreement based on the distributors track record with similar films.

The final option is the most complex (figure 3.5). It requires structuring the auction in a way that maximizes the owner's revenue. This format is better for more sought after assets.

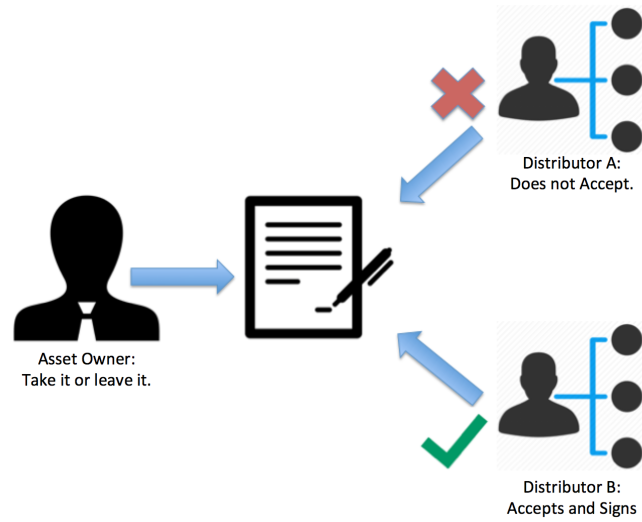


Fig. 3.3 Take it or leave it model. [85] [25] [45]

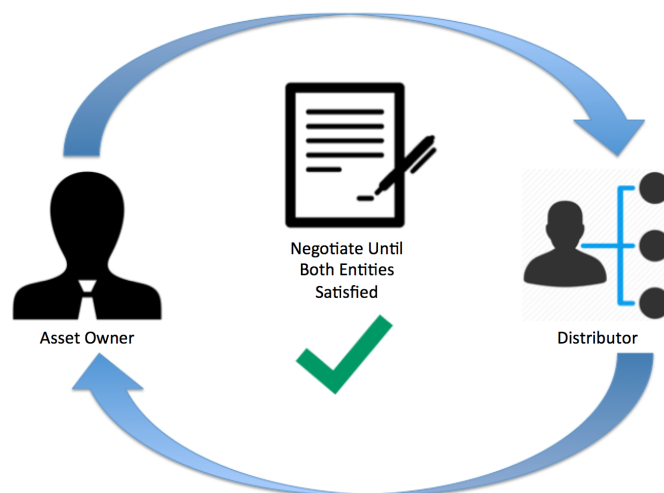


Fig. 3.4 Individual Negotiation Model. [85] [25] [45]

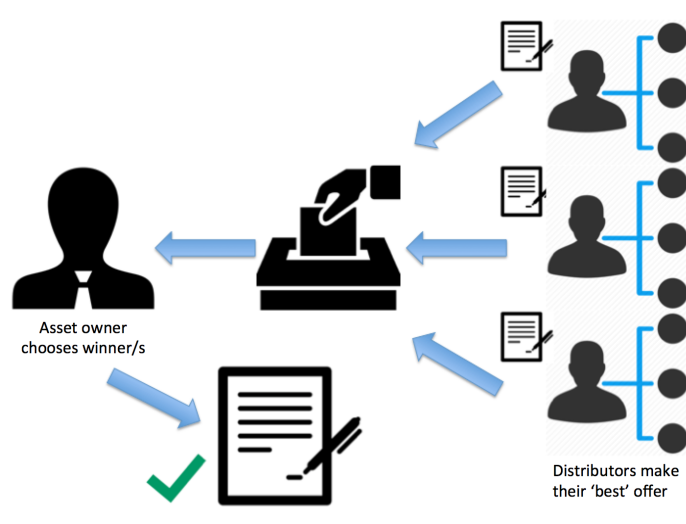


Fig. 3.5 Closed Auction Model. [85] [25] [45] [24]

If there are only 5 entities willing to distribute the asset it may be better to resort to one of the other options. However, if there are thousands, negotiating each individual contract becomes very cumbersome and this technique is best.

3.2.3 Signing Contracts

Once entities are happy with a contract they need to sign it. Again we can consider multiple options:

1. Send an image of their actual signature.
2. Create a user account and click a check box agreeing with the contract.
3. Use digital signatures based on cryptographic principals.

The first option is the most intuitive and simple when it comes to signing a contract. A user would sign on a piece of paper, take a picture of it and then send the picture. This provides little security. How do we know what the signature should look like, it could have come from anyone. We then have to store the images which requires a lot more than the other options and every time the distributor wants to prove who they are they need to send a picture again. If they send two different pictures of the same signature this will need to be verified either by a human or using complicated algorithms from machine vision.

The second option is what is most prevalent in today's web applications. Entities create accounts and then engage in all functionality through the account. Terms and conditions have to be agreed to upon creation of an account which outline the details of the service.

Whilst this would provide all functionality we require, it presupposes a central entity storing user accounts and providing conflict resolution. Essentially it requires Entities to trust some central service to maintain their data and provide the service. Since we are trying to enable the transfer of digital assets without a trusted middle man, we chose against this approach.

The final option is the most complex but based on the background research on Smart Contracts in section 2.2 seems to be the best. Each user has a self generated address from their password and can then provide mathematical signatures proving they know the password to the address without disclosing the password. Bitcoin and Ethereum both make use of Elliptic Curve Digital Signature Algorithms described in section 2.1.2. We will be using these signatures because as previously mentioned they provide sufficient security until computing power significantly increases at which point we can switch to Lamport Signatures or similar. There are other alternatives, provided by e.g. RSA, however none with the same efficiency in representation and calculation as the elliptic curve solution.

3.2.4 Management

So far we have discussed how we may create, negotiate and sign digital contracts. We now move on to what needs to be enforced and how we will achieve it.

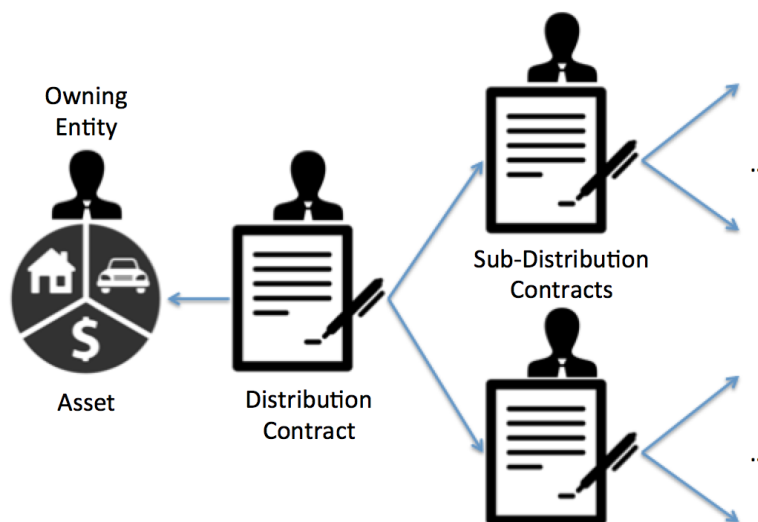


Fig. 3.6 Distribution Chain created using our definition in 3.2.1. [85] [25] [31]

The figure 3.6 illustrates an example distribution chain that could be created. This recursive structure allows complex relationships to arise between distributors and asset owners, giving users great flexibility in how they wish to distribute their asset. It becomes

clear when comparing with figure 3.1, that we are creating a new mechanism which gives users a greater incentive to spread digital content, allowing asset owners to reach a wider audience and generate higher profits than would otherwise have been possible.

Figure 3.7 illustrates what happens Whenever a user makes a purchase of one unit of the asset from a distribution smart contract. The cycle terminates when the root contract is reached, i.e. the distribution contract held by the owner of the asset. For each successful cycle money is split between the owning entity of the distribution contract and the parent contract.

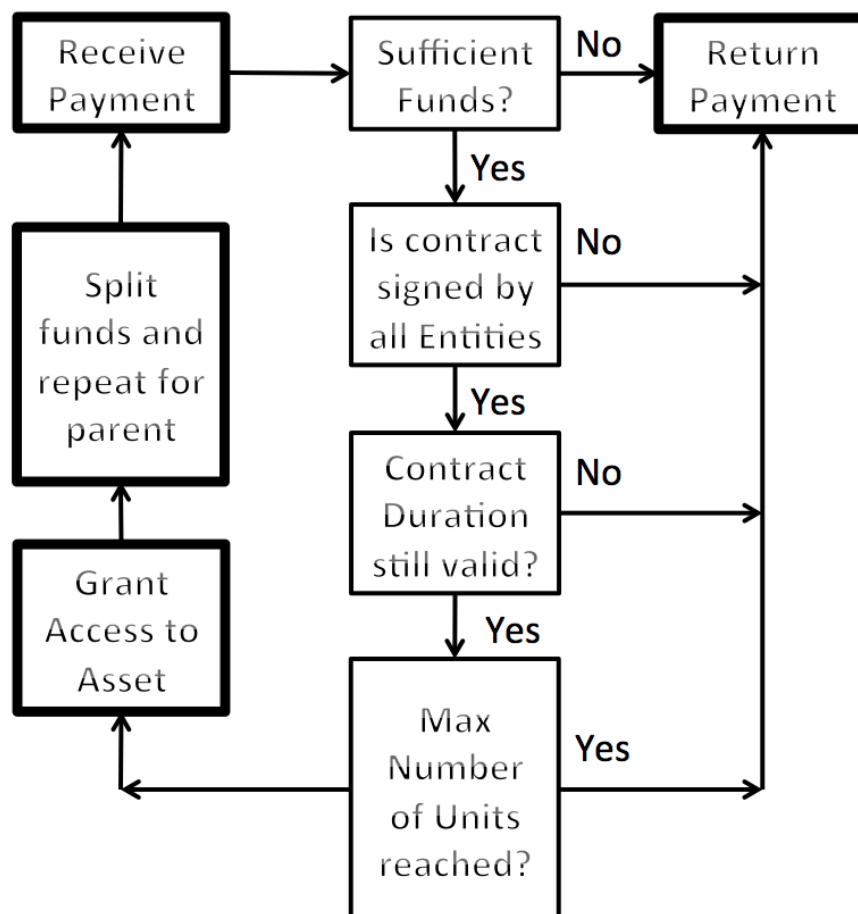


Fig. 3.7 Smart Contract Checks and Payment Flows.

3.3 Hierarchical Entities

As we have seen in the background research in section 2.6, current implementations are not looking at complex organizational structure for entities owning assets and associated payment flows.

Ujo, Bittunes and FilmFund all address this in so far as they allow the end user to purchase directly from the creator of the asset in a cryptocurrency and split the payment between a few owners at a predetermined fixed rate. We are looking to fill in all steps in between. With our model, distribution can spread naturally and virally, with all payments handled transparently and securely. This will be achieved by modeling the Entities in a hierarchical structure observable in today's companies.

The two problems we face here are the following:

1. How do payments reach all Entities? Do we have to keep track of every person that is owed some small amount?
2. How does a decision get made fairly for a company? Do owners decide? Do parent companies have a say in the decision?

The next subsections attempt to answer these questions.

3.3.1 Structure

Each entity needs to keep track of the following:

- A list of 'owners' (these can either be individuals or other entities).
- A mapping from owners to percentage owned. This will change every time there is a change in owners.
- The balance of money in the entity.
- A list of all assets owned and shared with the network.
- A list of all distribution contracts engaged in.
- A list of proposals outlining decisions the entity needs to make with the corresponding votes.

This structure is very beneficial from a users' perspective as they have complete control over their data. If they choose to close the entity, all associated actions and data are no longer

available to the other contracts. This is useful for incoming European legislation, mandating technology companies give users fine grained control over their data as described in the background section 2.4.

It is also worth mentioning that the Entity does not have to represent different owners. It could represent one owner with multiple addresses which has benefits on security and privacy. The owner can have multiple address which change over time allowing them to keep switching private keys as is good practice with any Smart Contract system as described in section 2.2 of the Background research. All funds then flow to different accounts making it harder for observers of the network to establish who the true owner of the addresses and entities are.

It is clear for the fields of and entity that it implicitly builds a hierarchical structure between individuals and entities as illustrated in figure 3.8.

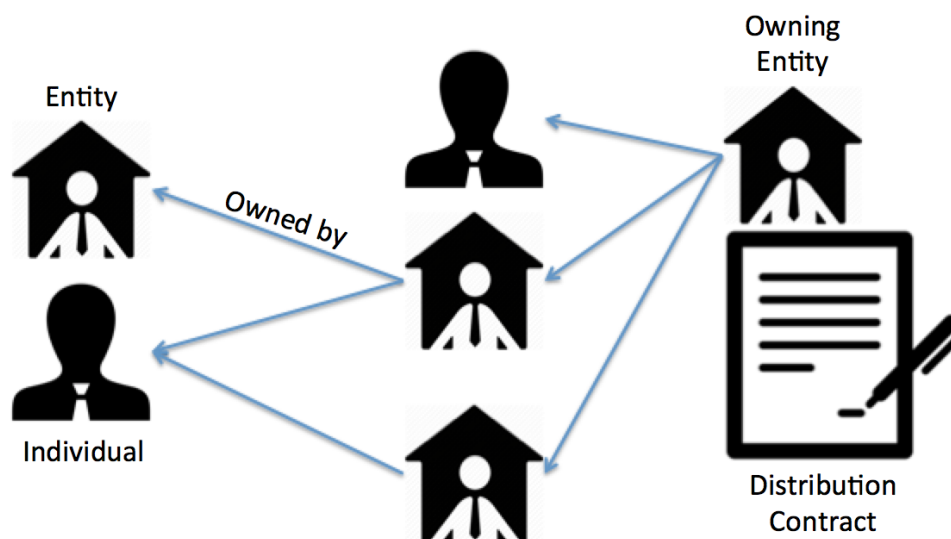


Fig. 3.8 Entities chain created using our definition in 3.3.1. [85] [25] [46]

This design allows efficient propagation of payments. Every time some user purchases some asset it eventually reaches at least one Entity as was show in section 3.2.4. Then once the payment reaches the 'root' Entity, it can be efficiently split based on ownership percentage to individuals and other Entities, which can then repeat the process until everyone has been paid.

It should be noted at this point that we will avoid cyclical dependencies i.e. where node A is node B's child and parent. This will result in infinite propagation of funds until an arithmetic underflow and is thus not stable.

3.3.2 Decentralized Decision Making

The next question is how to come to decisions between each owner of an entity e.g. deciding which distribution contacts to sign. We have multiple options:

1. Any owner of an Entity, no matter how small the ownership percentage, can make a decision on behalf of the Entity.
2. Every owner can receive a equal vote, and every decision is thus done democratically, by taking the action the majority voted for.
3. Every owner can receive a weighted vote, based on the percentage owned and then the action receiving at least 50% is taken.

The first case is clearly the simplest. It can be efficiently used when there are a small number of individuals associated with the entity. For Example consider a start up with 2 or 3 founders. It would not be unreasonable to assume that each founder would be entitled to make a decision on behalf of the company. The problem with individuals making decisions is that the other members may not know about them, which may lead to them to make conflicting decisions based on imperfect information. This could be extremely detrimental, making this option undesirable.

The second two options are both democratic and thus share the same advantages of being transparent and fair. The problem with every individual having the same voting rights is that this does not incentivize the listing multiple accounts for one user with a certain share. It also does not accurately represent how commerce functions in any industry and thus option 3 dominates option 2 for our intent and purposes.

It is important to note for the second two options that the voting must occur without a central trusted party. The behavior can be modeled nicely with Ethereum as described in section 2.2.5. An individual can make a proposition and a deadline, then each of the others involved in the entity can digitally sign it using the Elliptic Curve Digital Signature Algorithm outlined in section 2.1.2 along with their position. Once the deadline has passed, the action the Entity decided on as a collective will be taken.

At this stage we felt it to be beyond the scope of the project to also allow individuals to transfer their voting rights to others within the Entity or possibly even externally. We are also only creating one type of owner i.e. only one share class. In future we will consider different share types, some with voting rights and others with preferential dividend payouts.

3.4 Secure Storage of Digital Assets

It became evident in the Background section 2.2.5 that all information on Ethereum is publicly visible as all transactions are part of the Blockchain, this includes contract state. This means we will not be able to save any sensitive data that is not encrypted and even if it is encrypted the keys will have to be stored elsewhere. Also, the Ethereum network is not designed for storing large amounts of data since all op codes using persistent storage are expensive.

So the question becomes how to securely store users' digital assets. Given the research in section 2.3 we believe we have the following options:

1. Central storage on a private cloud based server controlled by us.
2. Storage on a Peer-to-Peer network controlled by no central party.

3.4.1 Centralized

The first option we have is to use the standard client server model as illustrated in figure 3.9. However this has the following problems due to the central server:

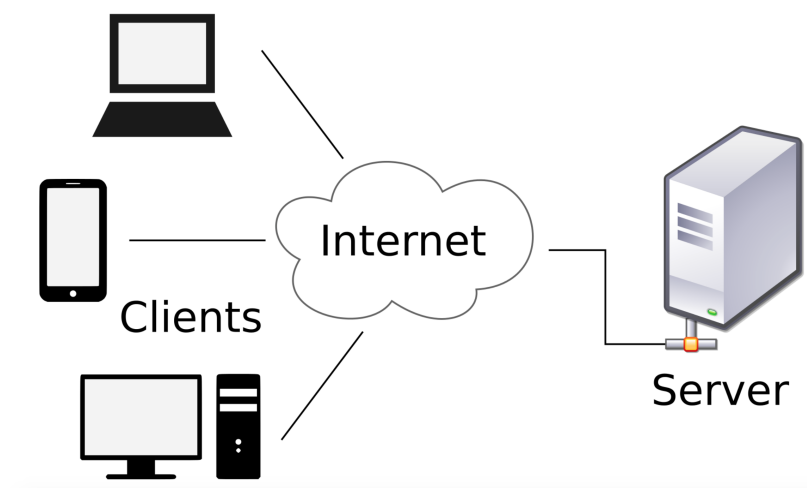


Fig. 3.9 Client Server Model. [88]

- If the server is successfully hacked, all assets are compromised.
- If the server is spammed by many requests, like in a denial of service attack, it crashes and our users are left without access to the assets until rebooted.

- Servers hosted in the cloud can scale efficiently, however the pricing is typically done in tiers. These tiers vary in pricing based on a maximum allowed bandwidth and storage usage amongst other factors. This means that if you do not fully utilize what you have paid for in a month, you just lose it which seems inefficient.

All three of these problems can be avoided using decentralized storage platforms.

3.4.2 Decentralized

As discussed in the Background research section 2.3.6 we have a choice between IPFS and Storj at the time of writing as the alternatives are not far along enough for our requirements.

For our purposes there is little difference between Storj and IPFS beyond the fact that Storj costs to use and IPFS does not directly. This is because Storj sends pieces of an uploaded file throughout the network as illustrated in figure 3.10 to be stored on other users' computers, the uploader does not need any local storage. IPFS stores the file on your local node and then gradually other users in the network become seeders as they access the data in the traditional peer to peer model illustrated in figure 3.11. Due to the cost difference we will use IPFS in this project.



Fig. 3.10 File on Storj split between many computers. [76]

Our assets are still not secure though. If we were to store the private key given by Storj or the hash given by IPFS in Ethereum, these would be publicly visible on the Blockchain and therefore accessible by anyone. A simple work around is to store the encrypted data on these platforms but then we cannot store the keys in Ethereum so how would a validated user gain access to the asset? Another problem to consider is even if a validated user was allowed to access an asset once they may not be in future so the key they receive needs to be a one off.

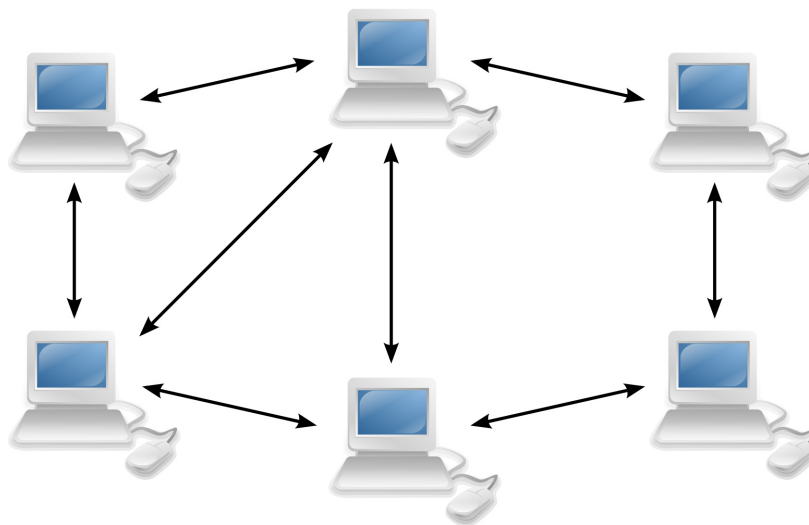


Fig. 3.11 A Peer-to-Peer Network. [2]

We only really have one viable option at this point which is to have a central server store the keys and handle the encrypting and uploading to IPFS as well as fetching, decrypting and streaming the content to a validated user.

There is an ambitious project at the Massachusetts Institute of Technology called Enigma which uses Secure Multiparty Computation and Secret Sharing Schemes in an attempt to solve this problem, however an application does not yet exist and is thus not an option for this project [91].

3.5 Putting it all together

Figure 3.12 illustrates the Smart Contract system we will be creating on Ethereum. There will also be a private server controlled by us, which will be used to encrypt and store the keys for the assets upon upload, save the file on IPFS and retrieve, decrypt and stream the film to users, validated by the system on Ethereum. Further details are available in the Implementation chapter.

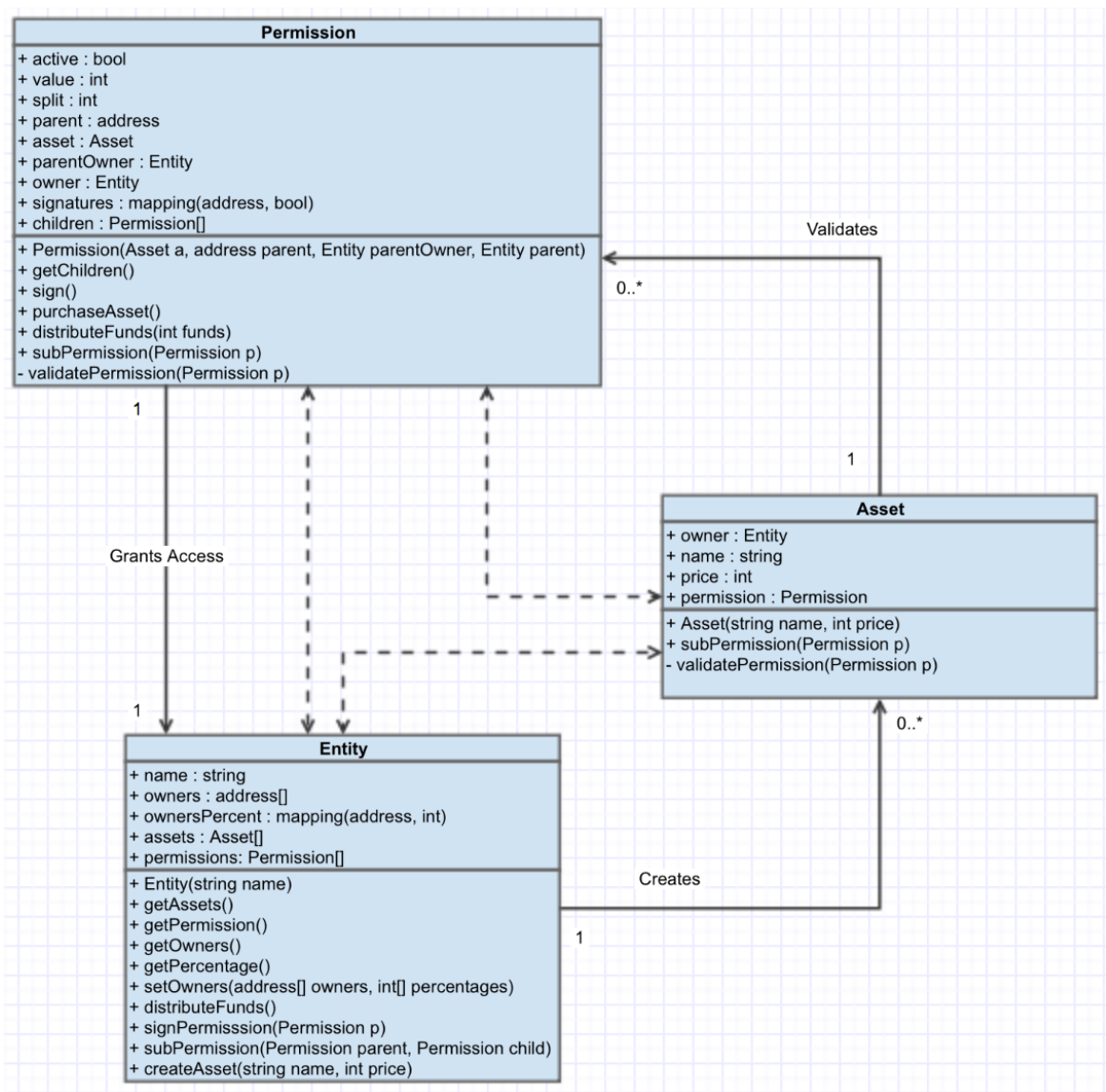


Fig. 3.12 Unified Modeling Language Diagram of System.

Chapter 4

The Value Chain

4.1 Overview

In this chapter, *The Value Chain* is introduced. We depict the front-end of the web application and discuss how it works from the user perspective for three main types of users identified for this product:

- **The Film Maker:** this user is the most important, at least in the initial usage growth phase of the website, since he is responsible for populating the application with assets, thereby attracting the other two user types. His objective in using the website is to open a new distribution channel for his films, namely micro-distribution via e.g. social media influencers enabled by Blockchain technology.
- **The Film Distributor:** this type of user is necessary for generating large amount of sales of the film. In the early stages of production, the film distributor finds the website via referral: either from the film maker, or by another film distributor. The film distributors objective in using the application is to generate revenue by entering into permission contracts with film makers or other distributors and sharing her permissions' corresponding view pages with consumers to generate views and hence revenue for all involved parties.
- **The Film Consumer:** this type of user is essential, but has limited interaction with the web application. The consumer's objective in using the application is to watch a film: he finds the watch link via promotion by one of its distributors. He proceeds to buy views of the film with his Ethereum account, thereby generating revenue for the distributors and film maker. He can then "spend" these views by streaming the video in the browser.

Ultimately, a user can and will most likely be a mixture of any of these three types, but the front end was designed in such a way to enable the simplest journey for each of these three different objectives.

Screen shots of the different pages will mainly be depicted in this chapter, but the mock-ups designed in Sketch for each page are shown in the Appendix.

4.2 The Film Maker

The most comprehensive use of the website corresponds to the film maker case, since access to the asset itself and its related permissions are mainly controlled by him. A diagram of this user journey is demonstrated in figure 4.1.

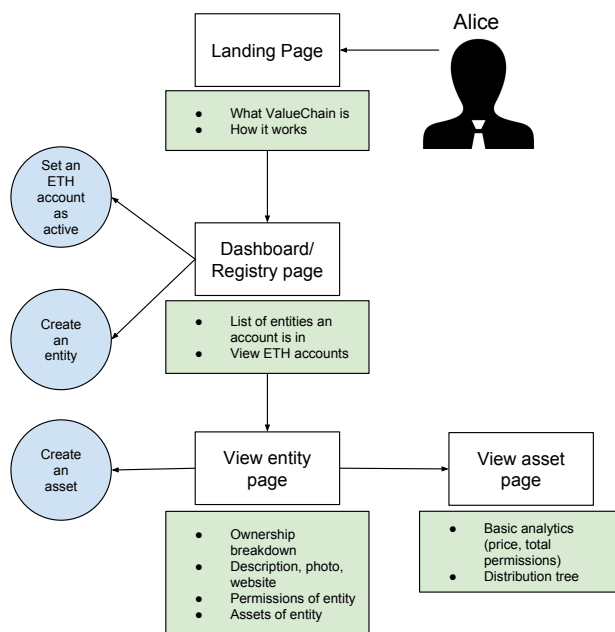


Fig. 4.1 The film maker's journey through the web application.

The film maker enters the website via the landing page, depicted in 4.2. Assuming he is using the Mist browser, we pull his current Ethereum account and display it in the navigation bar. It should be noted that this is the only type of user entering the website through the landing page, and hence the content is tailored to him (e.g. the heading states "reach a wider audience for your film". The rationale for this decision is that he is the most important user type: without the film makers, distributors and viewers will not need to access the website, so

it is important to primarily seek their traffic. The term "Blockchain technology" is mentioned a few times, most notably in the page header, to capitalize upon the media attention.

From here, the user clicks "start now" and is directed to the registry or dashboard page, a screen shot of which is demonstrated in figure 4.3. Here, the film maker can set any of his Ethereum accounts as active, view the entities each of his accounts owns, and create a new entity from that account. After an initial round of user feedback, we decided to put an explanation of what an entity is and why it is necessary in the "new entity" form modal because our sample of film makers had trouble understanding its purpose.

A user can find more information about an entity by clicking on the thumbnail, leading to the view entity page as depicted in figure 4.4. Since other entities (owned by one account) can be owners of a given entity, one can see an interactive pie chart of the ownership breakdown. Users can also see a list of assets owned by the entity, and a list of permissions. As can be seen in the Appendix, the original mock-up called used the term "permission" rather than "distribution contract". However, initial user feedback, claimed that the term "permission" is unclear, and hence it was replaced. From here, if a film maker's entity owns a portion of the one he is browsing, he has the option of "distributing funds" - e.g. splitting the balance of the entity, displayed in the header, to its owners.

From here, as explained in figure 4.1, the user can create a new asset and view an asset, the latter of which is illustrated in figure 4.5. From this page, a user can visit the watch page corresponding to the permission at the root of the distribution tree. Although the functionality is outside the scope of this project, there is also a button to an "analytics" section for the asset, which will in the future consist of data regarding the distributors of the asset: for example, how many views each of them sold or what portion of their revenue was generated by their sub-distributors rather than their own direct sales. It will also allow the film maker to track the performance of their film over time: their earnings over time, views over time, and if possible a breakdown of viewer demographics.

On this page, the film maker can see a real-time, interactive distribution tree for his asset. Each link between nodes represents a permission - the child node is the entity owning the permission, and the parent node is the parent owner of the permission. Upon hovering over the node links, one can see information about the permission: its address, which can be looked up in the Blockchain (see figure 4.6), its balance in ETH, and the proportion of funds that go to both the parent and child entities. If another user buys a view under one of the permissions of the asset, the tree updates in real time: the entity balances change as the revenue from the view trickles up the tree.

If the user clicks on the Value Chain logo in the navbar, he is redirected back to the registry/dashboard page depicted above in figure 4.3. Although the film maker has access



Fig. 4.2 Value Chain's home page.

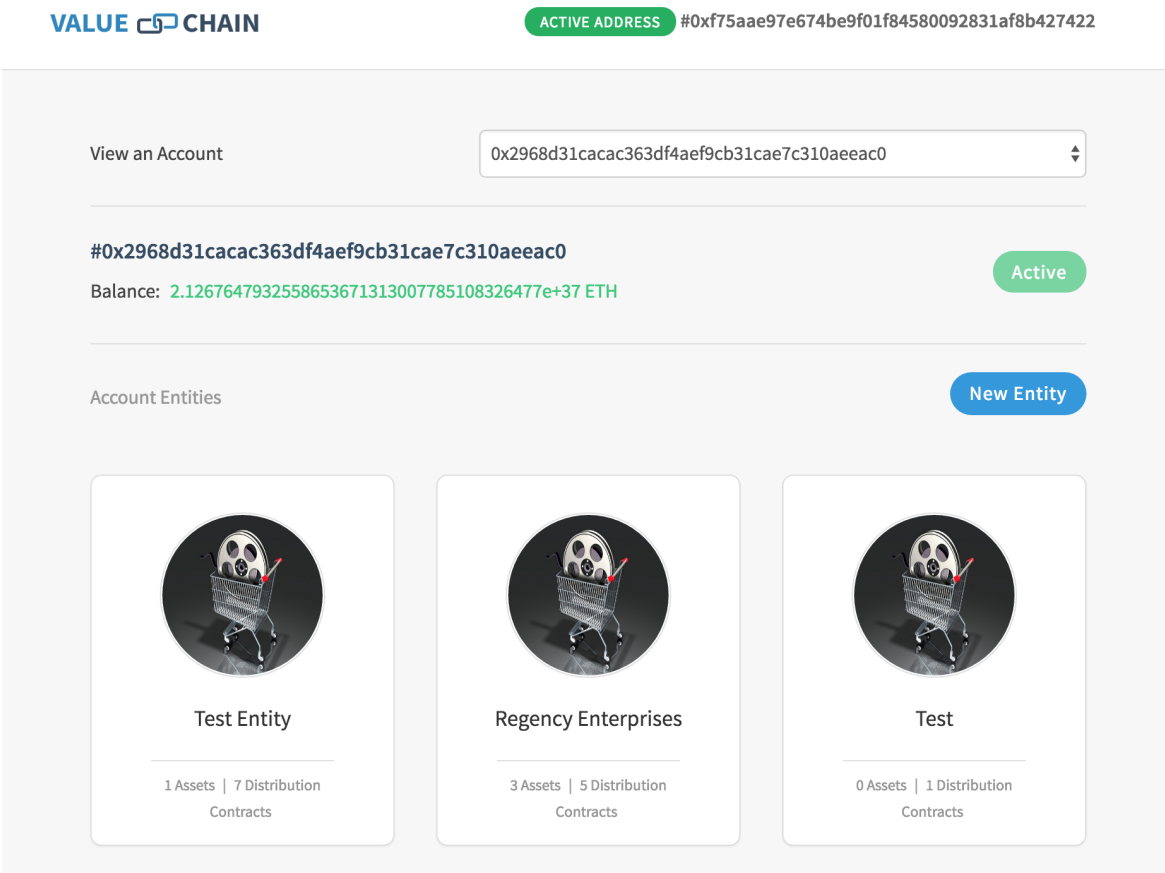

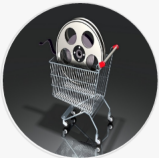





Fig. 4.3 A screen shot of ValueChain’s dashboard page.

VALUE  **CHAIN**

ACTIVE ADDRESS #0xf75aae97e674be9f01f84580092831af8b427422



Regency Enterprises


 0.00 ETH  regencyenterprises.com  Hollywood, CA

[Distribute Funds](#)
[New Asset](#)

About


Regency Enterprises is an American Los Angeles-based entertainment company formed by Arnon Milchan.

Ownership



- Universal Studios
- Mark Andreessen
- Alan Vey (You)


Assets



The Revenant

Shell is a global group of energy and petrochemicals companies...

Distribution Contracts



The Revenant

#0x1cb70833ac8e805fcd1e2c8934b0abb9ac198e9d


 [Regency Enterprises](#) (parent) and [Regency Enterprises](#)

Fig. 4.4 A screen shot of the view entity page.

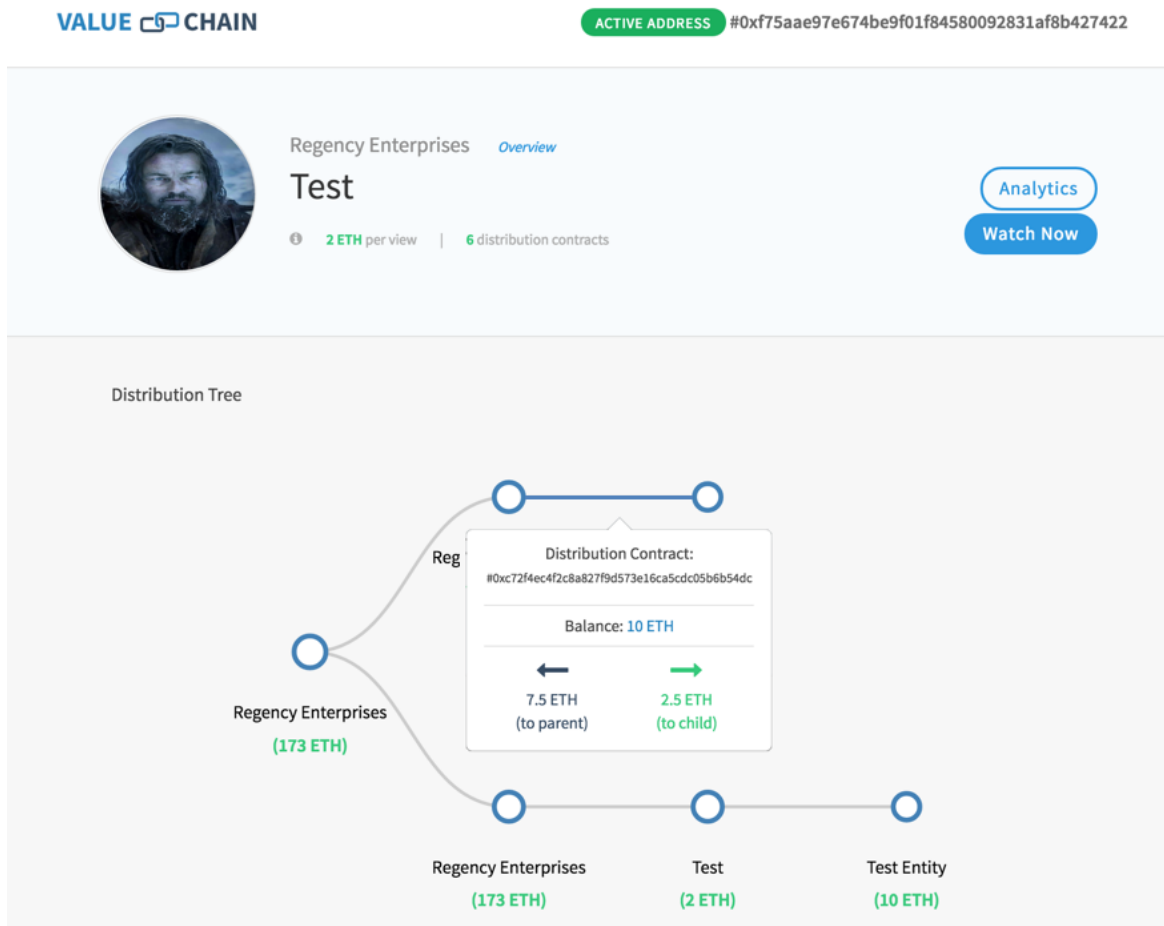


Fig. 4.5 A screen shot of the view asset page.

```

Contract created: 0x2ead0838fc15c5e233c163edf0ed0e1053fc555f
Gas usage: 1156976
Block Number: 0x6f

eth_newBlockFilter
eth_getFilterChanges
eth_getFilterChanges
eth_getTransactionReceipt
eth_getCode
eth_uninstallFilter
eth_sendTransaction

Gas usage: 64675
Block Number: 0x70

eth_call
eth_call
eth_call
eth_call

```

Fig. 4.6 An example of a permission contract being mined and placed into the Blockchain.

to other pages, such as a permission view page (depicted in the subsections below), these are not an important part of his user journey since his primary concern is his entities and corresponding assets and their performances.

4.3 The Film Distributor

Next, we will discuss the user journey of a film distributor, a diagram of which is illustrated in figure 4.7.

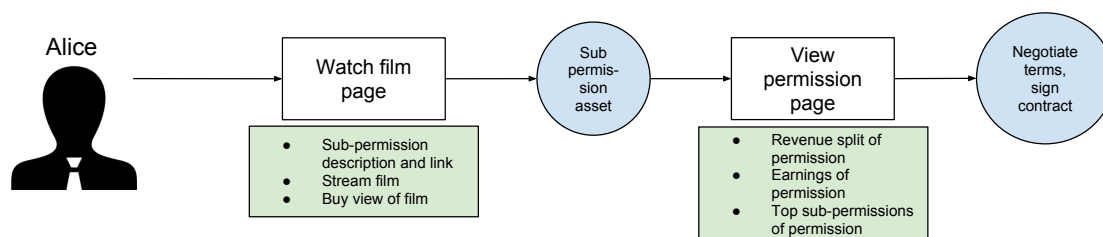



Fig. 4.7 A diagram of the user journey of a film distributor.

A first time film distributor user does not enter the website via the landing page and dashboard/registry because she cannot gain access to an asset or permission unless she already has an entity that owns these. Hence, her entrance to the website is the watch page, demonstrated in figure 4.8. She could receive this link by either a distributor that has offered her a sub-distribution opportunity (outside the realm of this application), or by the film maker asking her to distribute his film.


The user can then press the button "sub-distribute", which scrolls to the screen shown in figure 5.4. If one looks at the original design of this page in the Appendix, the original button was called "sub-permission", and there was no information about how the sub-distribution works. However, after user feedback we realized the distribution concept was novel and often misunderstood, so users requested more information to be displayed.

After reading this information and considering the film, if the user is interested in sub-permissioning, she can click "start negotiating", which currently only allows her to choose an entity to sub-permission from and redirects her to an inactive permission page for the default distribution contract of a revenue split 25%/75% in the parent distributor's favor, depicted in figure 4.10.

Future work will include re-directing her to a forum-like page, where her and the parent distributor can discuss the terms of the contract and come to an agreement. Once this happens, she will be re-directed to the inactive permission page with the terms they agreed upon.




VALUE  **CHAIN**

ACTIVE ADDRESS #0xf75aae97e674be9f01f84580092831af8b427422



The Revenant 2

★★★★☆

6 Views Remaining [Buy a View](#)

While exploring the uncharted wilderness in 1823, frontiersman Hugh Glass (Leonardo DiCaprio) sustains life-threatening injuries from a brutal bear attack. When a member (Tom Hardy) of his hunting team kills his young son (Forrest Goodluck) and leaves him for dead, Glass must utilize his survival skills to find a way back to civilization. Grief-stricken and fueled by vengeance, the legendary fur trapper treks through the snowy terrain to track down the man who betrayed him.

[Stream Now](#)

[Become a Sub-distributor](#)

Fig. 4.8 A screen shot of the watch page.

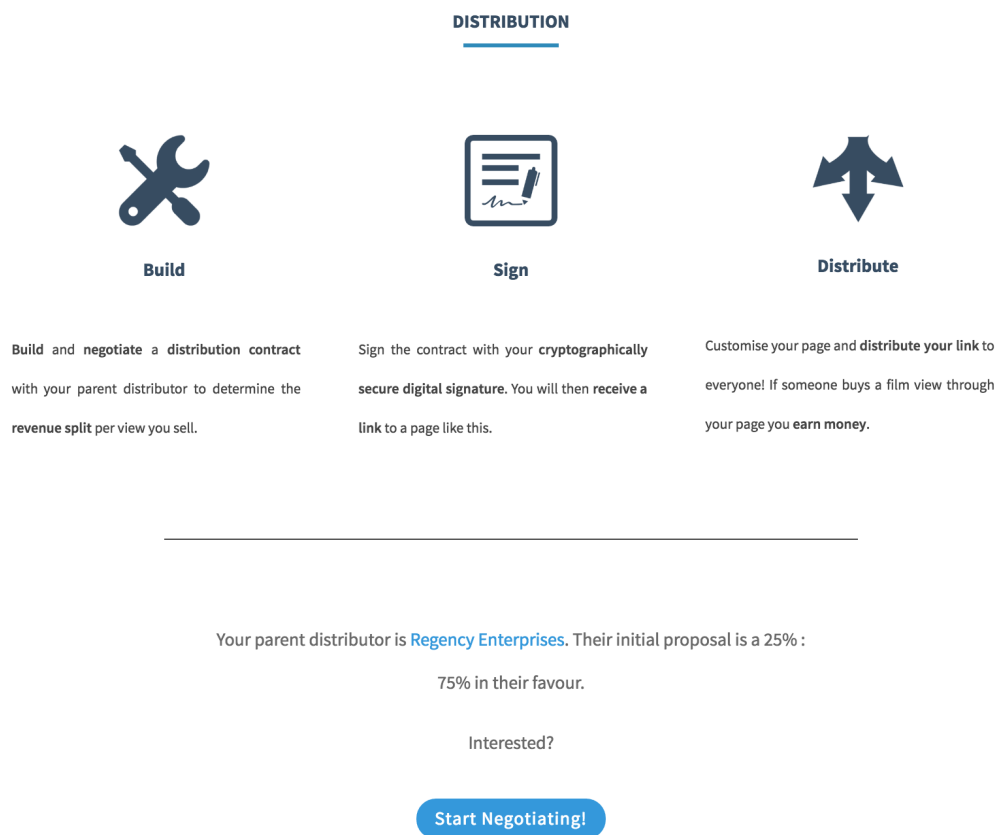


Fig. 4.9 A screen shot of the distribution section of the watch page.

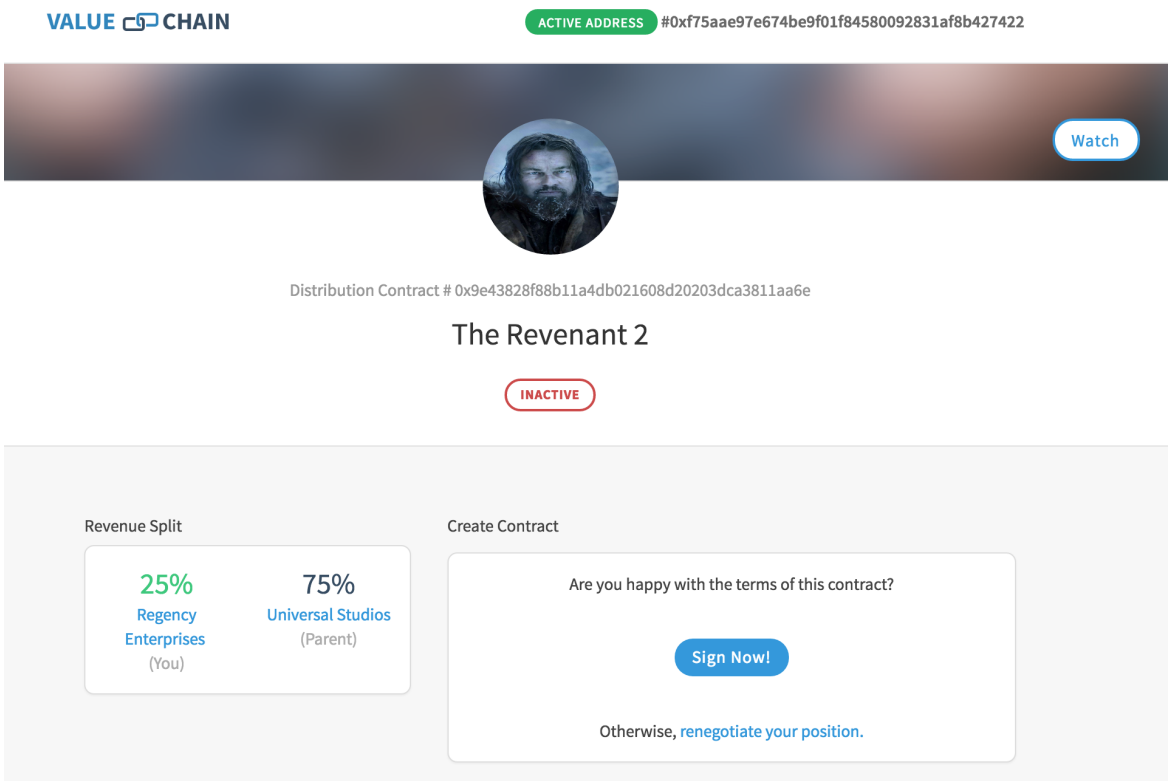


Fig. 4.10 A screen shot of the inactive default permission page.

Now, both the film distributor user and the parent distributor entity need to click "sign now", which simply opens a modal where the user selects which entity to sign from. This makes the permission "active" and redirects the user to the active permission page illustrated in figure 4.11.

It should be noted that one can click on "watch" to be re-directed to the permission's watch page, but no purchasing, streaming or sub-permissioning can occur until the permission becomes active.

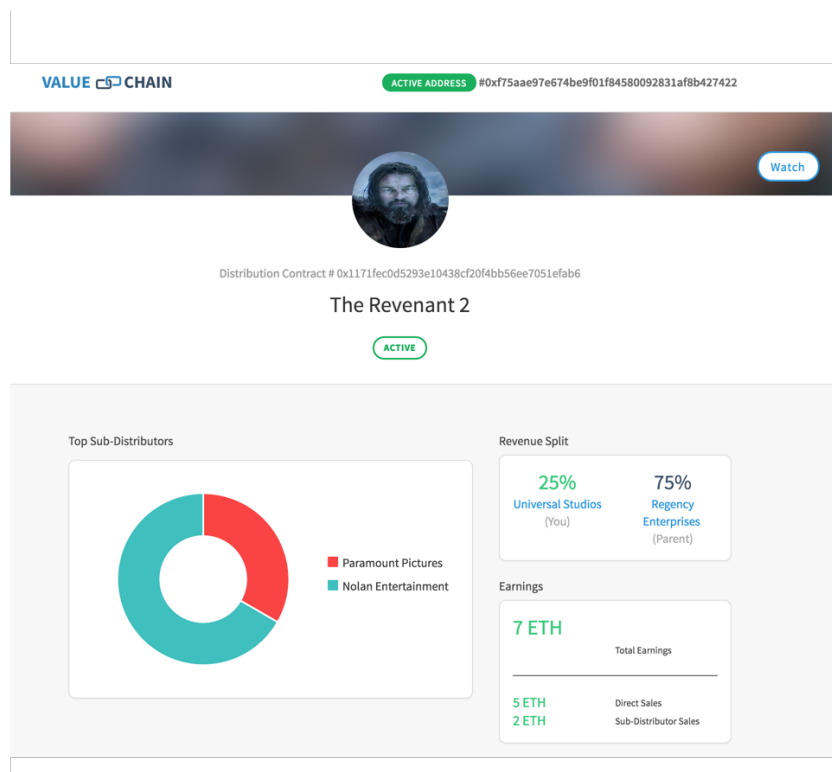


Fig. 4.11 A screen shot of an active permission page.

This page shows the terms of the permission contract along with data regarding the performance of the permission itself; its total earnings, along with the portions generated from direct sales (e.g. consumer purchased views from its own watch page) and sub-permission sales (e.g. views purchased from sub-permission watch pages). It also shows an interactive doughnut chart of the top sub-distributors and each of their contributions to this permission's revenue.

To distribute the film, the user must link the Watch page to potential customers and encourage them to buy views from it. Furthermore, once the distributor has created an entity and signed permission contracts, she now has a reason to visit the registry/dashboard page to manage her contracts and payment flows.

4.4 The Film Consumer

Lastly, we explain the user journey as a film consumer, as demonstrated in figure 4.12. This user almost exclusively interacts with the application from the Watch page (figure 4.8, as shown in the diagram). This is because he has no reason to create an entity or view asset and permission pages, since he can use his Ethereum account to buy and consume film views.

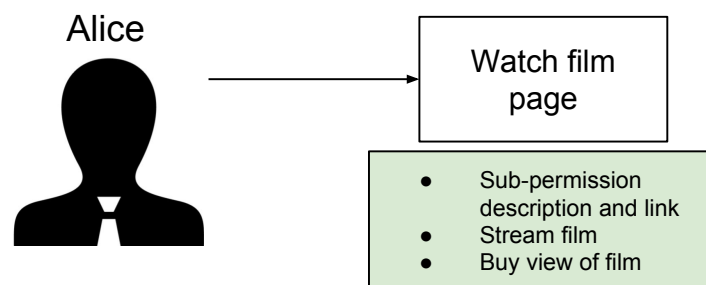


Fig. 4.12 A diagram showing the user journey of a film consumer.

Here, the user can purchase views of the movie, or stream the film, which produces a modal containing a high quality version as depicted in figure 5.16.

The film viewer could of course decide to sub-distribute the film as well, in which case his user journey becomes analogous to that described above in section 4.3. As stated previously, these user types are not mutually exclusive; a film consumer can also be a distributor, and a film maker can both watch and distribute other films on the Value Chain.

For future work, after the initial phase where accumulating film makers is the primary objective to grow the user-base of the site, the landing page will be re-worked to explain the incentives of using the Value Chain from a distributor and even film-maker's point of view. Furthermore, an "explore" section, which lists all of the films on the website allowing consumers to browse and buy views for any of them will be added to incentivize larger amounts of purchasing.

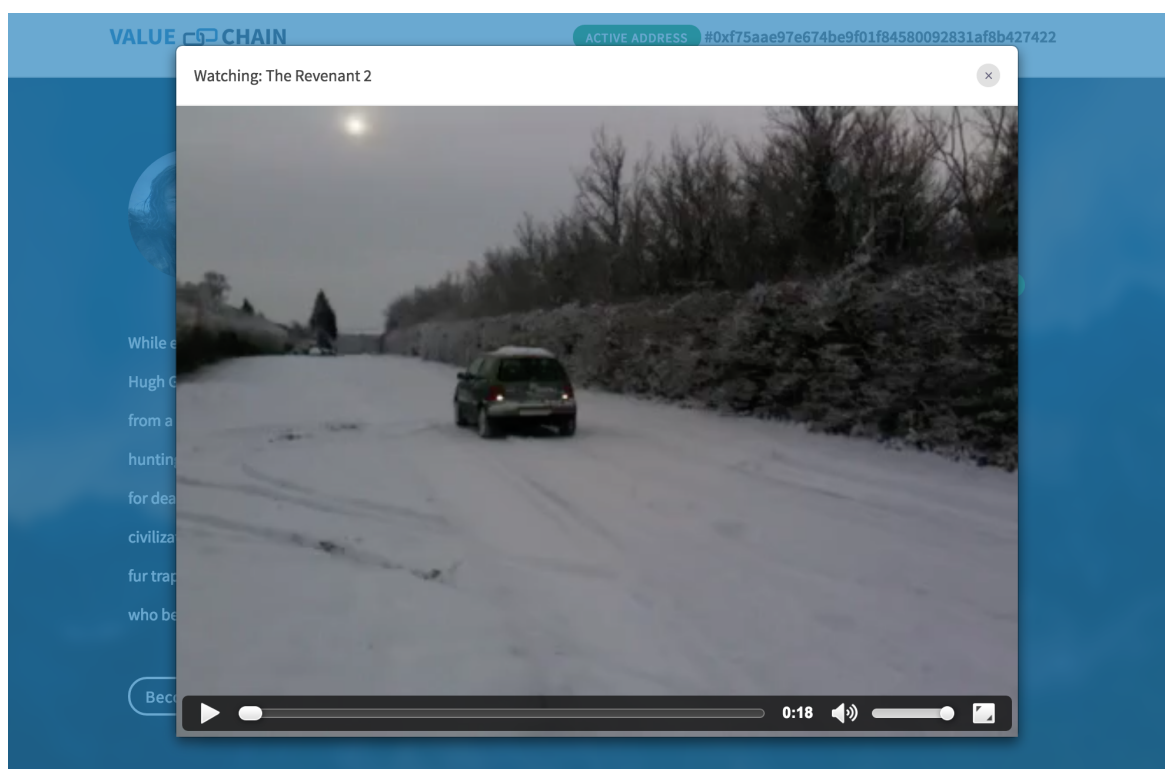


Fig. 4.13 A screen shot of the video streaming modal.

Chapter 5

Implementation

5.1 Overview

In this chapter we present the implementation details of our system. We will cover the overall system architecture, the core Smart Contract system, the private server processing assets and the front end application. We discuss the communication between each of the components and draw attention to some of the interesting problems faced during development.

5.1.1 Architecture

The value chain application is composed of three distinct sections: the Smart Contracts providing the core functionality for payments and permissions management, a private server handling sensitive information and the front end decentralized application allowing users to access the service, all of which is illustrated in figure 5.1.

A user, in this case Alice, connects to either a local IPFS node or one of the publicly available ones through the internet which in turn contacts the IPFS node with the front end files Alice is looking for. These in turn handle all communication with a local Ethereum node and the server.

As described in the Background section 2.2.5, the local Ethereum node contains a complete copy of the Ethereum Blockchain and a complete copy of the logs, where events are stored. It is connected to a variable number of peers over the internet. Our Entity, Permission, Assets and Registry (mapping from Ethereum Accounts to Entities) contracts are deployed to the Ethereum network and accessible using the web3 API to communicate with any local node using RPC calls. These calls are all handled on Alice's behalf by the front end files.

If Alice were to purchase one unit of an asset, the associated distribution contract (i.e. a specific Permission) fires an event which has its hash recorded in the transaction which is stored on the Blockchain, and the event is stored in the Ethereum event log.

The private server handles only functionality and information that cannot be made public without compromising the security of the system. All data is stored encrypted in a PostgreSQL database. The server also runs a local Ethereum node and listens for events granting users access to Assets and saves them to the database.

If Alice were to:

- request to view/stream the asset, an AJAX request is made to the private server and if Alice's digital signature checks out, the content is fetched from IPFS, decrypted and sent to Alice. The validation process involves checking the events in the database relating to Alice and the specific asset.
- upload a new Asset to our system, it would be sent to the private server using AJAX over HTTPS to ensure no one can intercept the data whilst being transmitted between the two endpoints. The server encrypts the asset and saves it to a local IPFS node. The keys are encrypted and stored in the database.

The following sections go into detail for each of the three main components of the application.

It should also be noted that we used the Git version control system. Version control is clearly essential to the development of any software application and because of the benefits Github offers to students with respect to private accounts, it was our clear choice. Imperial College's Department of Computing encourages students to use Git as they provide their own equivalent of Github, so I suppose my choice is based on familiarity rather than having analyzed alternatives in detail.

5.2 Smart Contracts

5.2.1 Infrastructure

As discussed in section 2.2 of the Background research we decided to use Ethereum as our Smart Contract platform, primarily because it has the second largest developer community in this field (second to Bitcoin) and the RootStock system being developed will allow hosting contracts written in Solidity to run on a Bitcoin side chain rather than the Ethereum Virtual Machine should this be beneficial in the future. Another benefit is that the project is completely open source enabling us to understand fine details in its implementation.

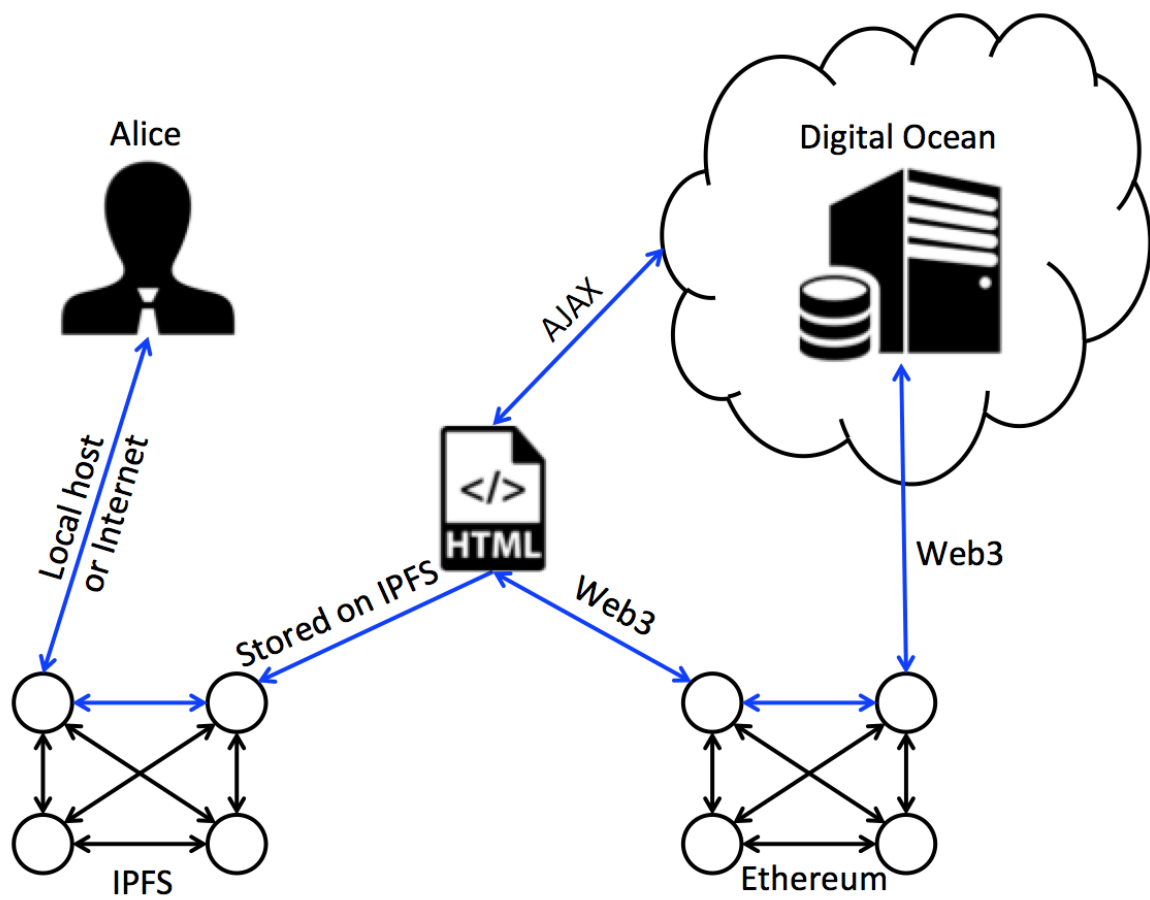


Fig. 5.1 Running all Smart Contract Tests. [48] [49]

Ethereum has a large developer community and provides many resources to aid quick creation of smart contracts. There are 3 official implementations of the client: one in C++, one in Go and one in Python. From a users perspective each provides the same functionality which is allowing RPCs over localhost:8545 by default through use of the web3 API. Each of these has a corresponding command line tool that can be used to access the Ethereum network, create accounts and contracts and interact with them.

Contracts can be written in two different languages:

- Solidity: Inspired by JavaScript, compiled to EVM op codes. We chose to use this language as Ethereum is currently focusing on it as its flagship language. Therefore the documentation is much more extensive, the inter-operability with the web3 API is better and the compile is more advanced, especially with respect to optimization.
- Serpent: Inspired by Python, compiled to EVM op codes.

We made use of the two user interfaced provided to ease the process of development:

- Mist: Decentralized application browser complete with Ethereum wallet. This browser allows users to use any application which makes use of the web3 API.
- Mix: Interactive development environment for smart contracts which simulates the Blockchain in memory for efficient local testing and can deploy completed applications to the Ethereum network. It also provides functionality to export the JavaScript required to create a contract through the web3 API along with the compiled binaries.

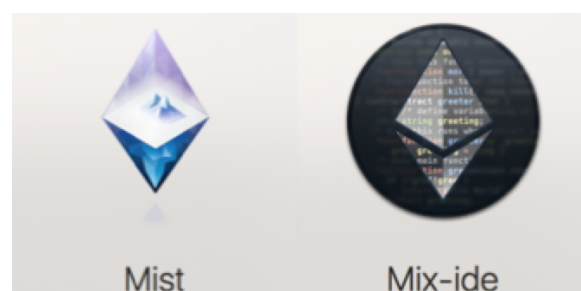


Fig. 5.2 Ethereum's Mist Browser and Mix IDE logos.

We set up a script that uses the command line compiler SOLC to compile the contracts and amend the JavaScript files in charge of creating them through the web3 API to speed up development for small changes to the contracts.

5.2.2 Contracts on Ethereum

Given the design of the Ethereum Virtual Machine and its Blockchain based operating model, implementing the Smart Contracts proved to be more difficult than initially anticipated. The contracts themselves are relatively easy to write and can be thought of much like objects in object orientated programming.

Creating a contract is done by using the web3 API and passing it the parameters for the constructor, the compiled code of the contract and enough gas to create it. This generated a transaction inserted into the Blockchain.

There are two operating modes for calling a method on a contract:

1. It can be run as a transaction, requiring gas and potentially some Ether to be sent along with the call. Any methods on a contract that mutate the storage (i.e. the persistent state) must compensate the network for the consumed space and computation so gas must be sent with it. The transaction becomes a part of the Blockchain and any changes are reflected throughout the network. When calling such a method through web3 we get the transaction hash returned even if the method should return some value as it has to be mined first. This introduces an average of 15 second delay (i.e. delay equivalent to average block confirmation time).
2. For means of efficiency, calls can also just be run locally. Methods that are just executing some calculations in memory and reading data do not effect other network participants. The return values of these functional are available immediately.

Because of these two different options I have taken great care to split code that mutates state from code that does not. This way a lot of the functionality of the contracts is accessible instantly and greatly reduces the cost associated with the system.

Also worth noting are some aspects to Ethereum we found unintuitive:

- When adding getters for fields that were declared as dynamic arrays we were expecting some difficulties as Ethereum needs to know the size of an array before returning it. The strange observation we made is that return an array of contracts (which under the hood is just an array of addresses) worked, however returning an array of addresses would only give the head of the array. In the end we managed to avoid any difficulties arising because of this by amending the design slightly.
- When methods are invoked on Ethereum contracts, the method has access to `msg.sender` and `msg.origin`. The origin is defined as the user that sent the original transactions. The sender is defined as the address of the contract or user invoking the method. However

if method on a contract invokes another method on that contract, we expected the other method to have the contract address itself as the sender. This is not the case though as the sender was still equal to the user. Again some slight adjustments to the design of the system allowed us to avoid this.

5.2.3 Entity

In section 2.4 we noted that incoming legislation in the EU mandates technology companies give users greater control of their data. In our system we give users full control over their contracts. All user information is saved in an Entity which can be destroyed only by the user. illustrated in figure 5.3.

```
function kill() {
    if(ownersPercent[msg.sender] != 0) {
        distributeFunds();
        suicide(owner);
    }
    else
        throw;
}
```

Fig. 5.3 Entity contract kill method implementation.

```
function distributeFunds() {
    if (ownersPercent[msg.sender] == 0)
        throw;
    else {
        uint bal = this.balance;

        for (uint i = 0; i < owners.length; i++) {
            uint share = (bal * ownersPercent[owners[i]]) / 100;
            owners[i].send(share);
        }
    }
}
```

Fig. 5.4 Entity contract distributeFunds method implementation.

This code fragment shows that only if the user invoking the method has an ownership percentage in the Entity, funds are distributed between the owners and the contract is closed sending any remaining balance to the caller as illustrated in figure 5.4. Funds are

merely distributed based on ownership percentage but in future we will add more complex functionality like precedence and periodic dividend payments. Important to note here is that owners is a list of addresses. These can be individual user addresses or addresses of other Entity contracts. This is how we enable very simple creation of and payment propagation between a hierarchy of Entities. We did not have sufficient time to implement decentralized decision making and thus any owner of an Entity can make decisions on behalf of all owners. More details are available in the evaluation chapter.

Thanks to the immutable Blockchain, even once a contract is deleted there is still a historical record of it so that our application does not end up with an inconsistent state when referencing a closed contract.

Since we would not like to force users to keep track of the addresses of all Entities they belong to we also created a Registry contract which does no more than provide a mapping from user account to associated Entities. The Registry will only ever have to be deployed once.

5.2.4 Asset and Permission

The Asset contract is simple, merely acting as a placeholder for all information relating to an asset such as the owner, the name and the price. It also keeps track of a permission. This permission can be considered as the root of the distribution chain. Therefore the asset must validate that any permission saved in this field references the asset itself, and lists the owner of the asset as both the owner and distributor of the contract as shown in figure 5.5.

Setting the permission in an asset requires validating it and then adding it to the list of permissions of the associated owning Entity as illustrated in figure 5.6. This also shows how seamlessly one contract may call methods on another.

```
function validate(Permission p) constant private returns (bool b) {  
    b = p.asset() == this &&  
        p.parentOwner() == owner &&  
        p.owner() == owner;  
}
```

Fig. 5.5 Asset contract validate method implementation.

The Permission contract represents one distribution contract, i.e. one link in the distribution chain. When a user wishes to purchase an asset we must first check if the contract has been signed, and that sufficient funds have been attached to the transaction to buy one unit of the asset. If this check finds a problem the funds are returned and the contract throws an

```
function subPermission(Permission p) {  
    if (validate(p)) {  
        permission = p;  
        owner.addPermission(p);  
    }  
    else throw;  
}
```

Fig. 5.6 Asset contract subPermission method implementation.

error. If it does succeed, we fire a Access event, specifying the sender of the message and the asset which is later picked up by the private server. We also distribute the funds between the parent permission and the owning entity and recursively do so all the way up the tree until we reach the root permission. The root permission then transfers all funds to its owning entity, the owner of the asset. Figure 5.7 shows our implementation in solidity for this process.

Due to the cryptographic primitives Ethereum uses as detailed in section 2.2.5 signing of the contracts is trivial. Any method invoked on a contract by a user, requires the user to submit a signature, generated using ECDSA. If the signature is not valid, the code is never executed so all we need to do in the method itself is insert a mapping from address to Boolean indicating which addresses have called the method. If both the owner and distributor have done so, the state of the contract is set to active as illustrated in figure 5.8. The method is declared constant as it does not mutate any state and can therefore be run in memory on any node without creating a transaction.

5.2.5 Testing

There are two main testing frameworks used for testing Smart Contracts. They are both aimed at creating DApps with asset pipelines and simplified contract development. They provide much more functionality than I require and use the following test frameworks at their core:

- Embark uses jasmine.
- Truffle uses Mocha/Chai.

For my purposes using a node package called testrpc which simulates Ethereum in memory combined with the testing framework jasmine was sufficient to unit test the Smart Contracts as is illustrated by one of the tests in figure 5.9. I chose Jasmine because it is used by Embark, the most successful development environment for Ethereum to date.


```
function purchaseAsset() {
    uint price = asset.price();

    if(active && msg.value >= price) {
        distributeFunds(price);
        Access(msg.sender, asset);
    }
    else {
        msg.sender.send(msg.balance);
        throw;
    }
}

function distributeFunds(uint funds) {
    if(this.balance >= funds) {
        value += funds;

        if(asset == parent)
            owner.send(funds);
        else {
            uint parentSplit = ((100 - split) * funds) / 100;
            uint ownerSplit = funds - parentSplit;

            owner.send(ownerSplit);
            parent.send(parentSplit);
            Permission(parent).distributeFunds(parentSplit);
        }
    }
    else throw;
}
```

Fig. 5.7 Permission contract purchaseAsset and distributeFunds methods implementation.

```
function sign() {
    signatures[msg.sender] = true;

    if(signatures[parentOwner] && signatures[owner])
        active = true;
}
```

Fig. 5.8 Permission contract sign method implementation.

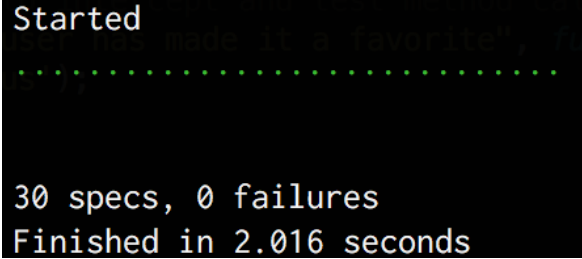
```

describe("A permission", function() {
  it("becomes active when signed", function() {
    createPermission("TestName", function(error, contract) {
      expect(error).toBe(null);
      expect(contract.address).not.toBe(null);
      expect(contract.active()).toBe(false);

      contract.sign();
      expect(contract.active()).toBe(true);
    });
  });
});

```

Fig. 5.9 Testing signing permission contract using Jasmine.



```

Started
.....
30 specs, 0 failures
Finished in 2.016 seconds

```

Fig. 5.10 Running all Smart Contract Tests.

5.3 Decentralized Application (DApp)

5.3.1 Infrastructure

Ethereum suggest users create their Decentralized Applications using the Meteor framework which is what we decided to do as this allows seamless deployment to the Ethereum DApp Store.

Meteor is used to create web applications is JavaScript (specifically we are using some of the new features from ES6). It is a framework that attempts to allow users to focus on implementing the application rather than dealing with the configuration and communication between the individual parts. It is completely reactive and works with HTML. It has an adapted version of Handlebars call Spacebars, allowing users to add some functionality to HTML. For example: variables set from JavaScript or looping over a collection adding a div for each item. All styling and formatting is done using Cascading Style Sheets.

Meteor has a very active development community and allows inclusion of libraries simply through a package manager called Atmosphere. We made use of the following packages:

- chart:chart: Used to create the pie charts visible on the permission page and the entity page.

- `d3js:d3`: Used to create the tree structure illustrating the distribution chain on the assets page.
- `ethereum:web3`: Used for all communication with the local Ethereum node to create and query contracts and individual users.
- `iron:router`: Enables front end routing using JavaScript. This allows us to render different templates based on what the URL is.
- `fontawesome` and `twbs:bootstrap`: Used to create reactive front end and styling.
- `jQuery`: Used to mutate the DOM and do AJAX calls to the private server.

Adding these packages using `meteor add` on the command line was very useful. Not only does it automatically download the files, it also imports them in the correct order. After using `meteor add`, all functionality from the packages is immediately available in any JavaScript or HTML file.

The following is a brief description of each of the pages we implemented combined with some interesting aspects to them. For a full description please see chapter 4 titled The Value Chain. Figure 5.11 depicts the typical paths users would take through the system.

- **Registry**: This is the root directory (i.e. "`<IPFS hash>/`"). It displays all accounts for the local Ethereum node running on `localhost:8545`. Each account can be set as active and the user can then create Entities from their active account.
- **Entity**: The directory is "`<IPFS hash>/entity?address=`". This page displays all information stored in an Entity contract and visualizes the data in a pie chart. It provides links to asset, permission and upload pages.
- **Upload**: The directory is "`<IPFS hash>/upload?entity=`". This page handles creation of an Asset contract and the upload of the asset to the private server. Once the upload is complete and successful it redirects to the asset page.
- **Permission**: The directory is "`<IPFS hash>/permission?address=`". This page displays all information stored in a permission contract and visualizes some of it using a pie chart. It allows users to sign the permission and links to the watch page.
- **Asset**: The directory is "`<IPFS hash>/asset?address=`". This page displays all information stored in an Asset contract. It also visualizes the distribution chain associated with the asset and links to the watch page.

- Watch: The directory is "<IPFS hash>/watch?permission=". This page describes the three options a user has available to them: The can purchase a view, stream the asset consuming a purchased view or become a sub distributor. If they choose to become a sub distributor they are redirect to their own newly created sub permission page.

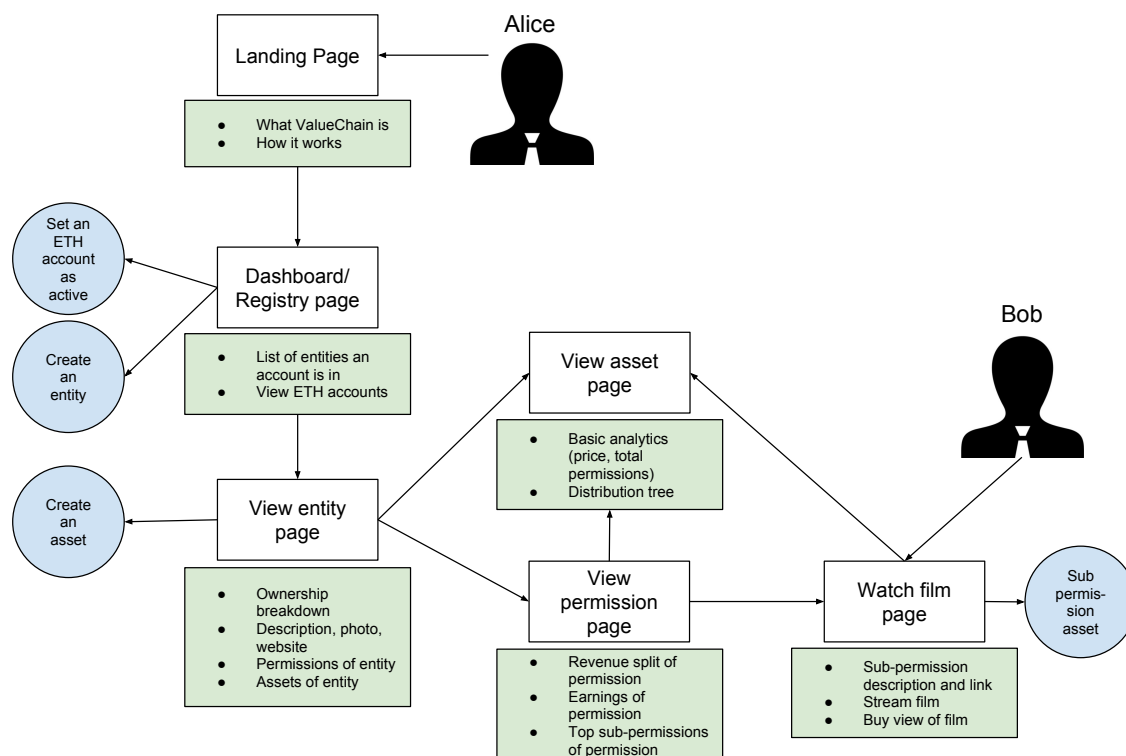


Fig. 5.11 User flow through the various pages of the application. [85]

It should be noted that no user accounts are required when building an application for Ethereum, because all interactions are based off of a users public key which they prove to be theirs by signing messages with their corresponding private key.

5.3.2 Integration with IPFS

NPM has a package call meteor build client which puts the static files into a build folder, allowing us to run the application by simply opening the build/index.html file. We uploaded this build folder to the IPFS network to allow anyone to access it.

The problem with uploading a file to IPFS is that it generates an immutable address because the address is partly the hash of the uploaded files. This means if we make any change to the application, the URL will change and our users will have to tell the new address.

To solve this problem we made use of the Inter Planetary Naming System (IPNS). This system allows generates you a URL that points to a different IPFS URL. This pointer can be amended as we make changes to our application to point to the latest hash. This way our users will only ever need one address and we can mutate our site whilst still being completely based on the IPFS network.

This process is illustrated in figure 5.12 .

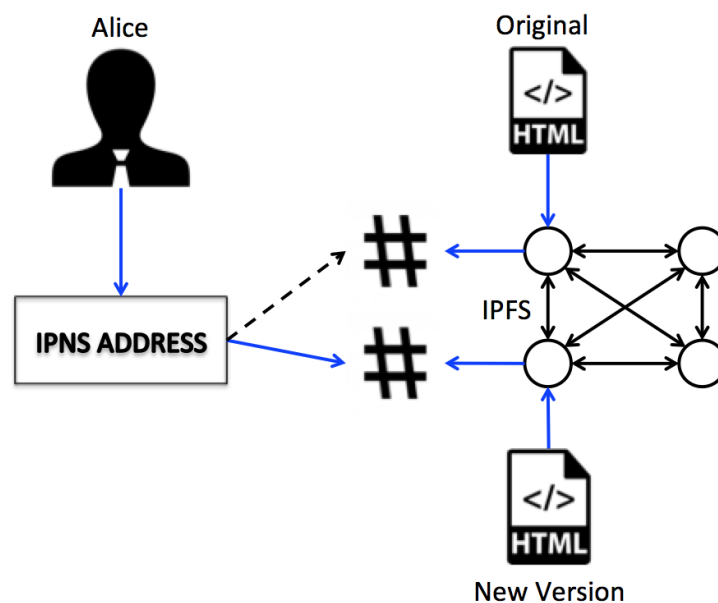


Fig. 5.12 Illustration of IPNS pointing to different IPFS hash addresses. [48] [85] [47]

5.3.3 Integration with Ethereum

In the previous subsections we have mentioned the Front End files pulling information from Ethereum. To use our site the user must be running an Ethereum node locally and have it accepting JSON-RPC on port 8545. An Ethereum node is easy for users to get they have two options:

1. Download Mist the Ethereum browser which itself creates an Ethereum node for the user. This is probably not going to be the main way our users interact with the site because switching browsers is annoying to them.
2. Downloading and installing the Ethereum C++, Go or Python bundles which will allow them to use the Alethzero client or other unofficial clients to manage their local node. This is how we expect the first users to interact with our site.

It should be noted that the Homestead Ethereum release is focused on the stability of the network for developers not end users. The next release is called Metropolis and describes itself as the release for the masses. This is where polished user interfaces and user experience will be promoted in all Ethereum DApps and tools used to access them.

5.3.4 Testing

We tested our Application using the functionality provided by Meteor. This can be tricky in some cases when dealing with the Meteor client server communication but since our information is stored in Ethereum we do not have a server component in the Meteor app.

We performed full application integration testing. A problem we came across when testing reactive components, which we have throughout our application, is that Meteor is eventually consistent on the front end but there is not guarantee of when it will update. Functionality such as `Tracker.afterFulsh()` was used to wait long enough in testing to ensure reactive changes occurred before testing for the results thereof.

We used the `dispatch:mocha-phantomjs` Atmosphere package which uses Mocha for testing with the Chai assertion language and runs the client tests within a PhantomJS page printing the results to the command line. Figure 5.13 demonstrates a sample test's source code.

5.4 Private Server API

5.4.1 Infrastructure

When coming to a decision about which technologies to user for our API we had multiple factors to consider as well as keeping the costs to a minimum.

First and most importantly we needed to be able to guarantee and prove the security of the system. The server will be handling all sensitive data that has value and thus needs to take great security precautions. The communication between users and the server is handled by HTTP over SSL so we have to consider all security measures once the data reaches the application. This includes encryption of the asset, validating of users' digital signatures and therefor encryption of the database.

The second factor is that this is the initial design of system and we will most likely be making many iterative changes. For this reason we need to use technologies with great developer communities and documentation to enable adding functionality quickly. This will promote an Agile development approach which should yield the best possible application.

```
// Ensures reactive components have rendered before testing them
const afterFlushPromise = Promise.denodeify(Tracker.afterFlush);

if (Meteor.isClient) {
  initialiseTestRPC();

  describe('Ethereum data is correct when', () => {
    beforeEach(() => FlowRouter.go('/'));

    describe('loading the root page', () => {
      it('has all accounts listed', () => {
        return afterFlushPromise().then(() => {
          var accountNum = $('#accounts').children.length;
          assert.equal(accountNum, 10);
        });
      });

      it('has all entities for the selected account', () => {
        return afterFlushPromise().then(() => {
          var entityNum = $('*[id*=view_link]:visible').length;
          assert.equal(entityNum, 5);
        });
      });
    });
  });
}
```

Fig. 5.13 Integration testing the applications root directory.

We decided to use Ruby on Rails because it is free and open source. Not only does it have a huge ecosystem with the package manager bundle which downloads gems (i.e. libraries), but it also allows for very efficient app development. The framework clearly uses the dynamic language Ruby which is very easy to read and understand through clever implementation of "everything that can be is an object, even class definitions". It has great gems for handling cryptography (some specifically for Ethereum) and provides great tools for testing the application. Alongside these benefits, we also have a lot of experience developing with Ruby on Rails on previous projects.

When choosing where to deploy our applications we identified two options: Heroku or Digital Ocean. Both are cloud computing companies which allow users to push an application to them using our version control software Git and immediately deploy them to an IP address which makes them exceptionally easy to use. Heroku's lowest tier is free and thus initially seemed like the best option. However once we realized we would not only need to deploy the application but also have a local Ethereum node running Digital Ocean became our top choice because it allows us to SSH into a virtual machine allowing simple fine grained control over the server. Digital Ocean's lowest priced tier is 5permonth,howeverwereceived a100 voucher for free as part of their student package.

Digital Ocean offers a one click rails options. This created a machine for us and installed and ran a sample rails application using Unicorn and Nginx on Ubuntu and a postgresSQL database. All we did was SSH into the server, replace the sample application with ours, set up the database configuration and restart the unicorn server at which point the site was available on the machines IP address. Installing Ethereum and having the node run in the background was made easy using the Ubuntu advanced package manager. The max CPU usage had to be limited to 20% for downloading the Blockchain because Digital Ocean has a excessive use policy and for 30% our process was terminated multiple times. Once the Blockchain was downloaded we dropped the CPU max usage for the Ethereum node down to 10% which was sufficient for our purposes. We made use of the Ethereum ruby gem to handle communication with the node.

We also created an IPFS node on the server which was trivial using the IPFS installation script and the ipfs ruby gem to handle communication with the node.

5.4.2 API

As touched upon in chapter 3 titled Design we only need a means of uploading an assets and a means of streaming that asset. This resulted in the following three API methods also illustrated in figure 5.14.

```
Rails.application.routes.draw do
  get "stream/:id/:pubkey" => "assets#stream"
  get "check/:id/:pubkey" => "assets#check_access"

  post "upload" => "assets#upload"
```

Fig. 5.14 Ruby on Rails API for our private server.

1. `assets.upload`: A POST method that contains the asset, the asset contract address and the owners public key in the submission form data. The server encrypts the uploaded asset and stores it on IPFS. It the saves the asset contract address and the IPFS hash in the encrypted database as illustrated in figure 5.15.
2. `assets.stream`: A GET method that takes as parameters the Ethereum address of the Asset Contract governing access to the Asset and the `pubkey` parameter which is a string containing the user's public key and a digital signature thereof. The server first checks that the signature is valid using the `ruby_ecdsa` gem, then it checks in the database of Ethereum events if an access event has fired with the asset hash parameter and the user's public key. If so the asset is retrieved from IPFS, decrypted and streamed

to the user as illustrated in figure 5.16. Once this method returns we have a callback that marks one access as consumed so that a user has to purchase another unit of the asset through Ethereum if they wish to access it again.

3. `assets.check_access`: This method performs the same checks as the `stream` method but does not return the asset but rather a Boolean as to whether or not the supplied parameters do check out. This is used in the front end to so users know if they have access or need to purchase more units.

```
# /upload
def upload
  asset = Asset.new(asset_params)

  data, key, iv, salt, pwd = encrypt(params[:asset].read)
  hash = ipfsPush(data)

  asset.key = key
  asset.iv = iv
  asset.salt = salt
  asset.pwd = pwd
  asset.ipfs_id = hash

  if asset.save
    render json: { success: true }, status: 200
  else
    render json: {
      success: false,
      error: "File failed to save on the server" }, status: 500
  end
end
```

Fig. 5.15 Implementation of upload method on server.

5.4.3 En/Decrypting Assets

We made use of the `openssl` Ruby gem which wraps the OpenSSL library as illustrated in figure 5.17. We use the advanced encryption standard 128 using cipher block chaining for encryption and decryption. Keys are generated from a password using PBKDF2. These are the latest standards which is why we chose them for securing our assets.

5.4.4 Database

We used a PostgreSQL database because of how the one click Ruby on Rails is set up in Digital Ocean as previously explained. Our database stores very sensitive information such as

```

# /stream/:id/:pubkey
def stream
  asset = Asset.find_by_address(params[:id])

  if !asset.nil? and validate_pubkey(params[:pubkey])
    data = ipfsFetch(asset.ipfs_id)
    dc_data = decrypt(data, asset)

    send_data(dc_data, :disposition => 'inline', :stream => true)
  else
    render status: 403
  end
end

```

Fig. 5.16 Implementation of stream method on server.

```

def encrypt(data)
  cipher = OpenSSL::Cipher.new 'AES-128-CBC'
  cipher.encrypt
  iv = cipher.random_iv

  pwd = 'exact process redacted'
  salt = OpenSSL::Random.random_bytes 16
  iter = 20000
  key_len = cipher.key_len
  digest = OpenSSL::Digest::SHA256.new

  key = OpenSSL::PKCS5.pbkdf2_hmac(pwd, salt, iter, key_len, digest)
  cipher.key = key

  encrypted = cipher.update(data)
  encrypted << cipher.final

  return [encrypted, key, iv, salt, pwd]
end

def decrypt(data, asset)
  cipher = OpenSSL::Cipher.new 'AES-128-CBC'
  cipher.decrypt
  cipher.iv = asset.iv

  iter = 20000
  key_len = cipher.key_len
  digest = OpenSSL::Digest::SHA256.new

  key = OpenSSL::PKCS5.pbkdf2_hmac(asset.pwd, asset.salt, iter, key_len, digest)
  cipher.key = asset.key

  decrypted = cipher.update(data)
  decrypted << cipher.final

  return decrypted
end

```

Fig. 5.17 Implementation of encrypting and decrypting data.

the keys for the encrypted assets. Because of these we felt it essential to encrypt the database. Fortunately we found a gem called `crypt_keeper` which allows us to do handle encryption effortlessly with the Rails framework as shown in figure 5.18. The gem uses AES-256 and PBKDF2 for pass phrase derivation similar to the encryption of our data as seen in figure 5.17.

```
class Asset < ActiveRecord::Base
  has_many :accesses

  crypt_keeper :key, :pwd, :salt, :iv => :aes_new, :key => 'redacted', salt: 'redacted'
end
```

Fig. 5.18 Implementation of database encryption.

We have two tables in our database as shown in figure 5.19:

- Asset: This stores the IPFS address of the encrypted asset and all information used to encrypt it.
- Access: This stores the events from Ethereum that grant users access to an asset.

```
ActiveRecord::Schema.define(version: 20160522104523) do

  # These are extensions that must be enabled in order to support this database
  enable_extension "plpgsql"

  create_table "accesses", force: :cascade do |t|
    t.integer "asset_id"
    t.string "pubkey"
    t.boolean "consumed"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
  end

  create_table "assets", force: :cascade do |t|
    t.string "address"
    t.string "key"
    t.string "iv"
    t.string "salt"
    t.string "pwd"
    t.string "ipfs_id"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
  end
end
```

Fig. 5.19 Database Schema.

5.4.5 Testing

The server was relatively simple with only three methods in the controller. Similar to our tests for the Smart Contracts, we used the JavaScript Jasmine framework combined with jQuery to test each of the API methods in an integrated fashion.

Chapter 6

Evaluation

6.1 Overview

The Value Chain is the first implementation of an application that stores a structured collection of digital items of value and securely and transparently regulates the access to them without a trusted third party. It is more advanced than the other implementations with respect to managing the flows of value between the users creating, distributing and consuming the collection of assets and it creates a completely transparent audit trail.

Blockchain is a very new field and there are no mature businesses in it, merely start ups and university researchers working towards creating the first production grade application with a large number of users. The only project that has gained significant traction in this space so far is Bitcoin. Because of this it has been difficult to provide a detailed analysis of our efforts in terms of other successful projects since we do not know if the projects we have referenced will actually be successful.

This section will start by describing the industry feedback we have received in an attempt to gain a greater insight into the usefulness and chances of success of this project. We will then have a look at each of the three major components in detail: the Smart Contracts providing the core of our innovative functionality, the Decentralized Application built to showcase our systems applicability to the film industry and the server API which handles information that cannot be public without compromising the entire system. Finally we will have a look at the strengths and weaknesses of this project in the context of the Background research.

6.2 Industry Response

Since Blockchain is such a new field, we felt this project could not be successfully evaluated without significant industry feedback. Part of the motivation for this project from the outset was to create an application that would actually be useful to the individuals in the film industry and then branch out into other industries once successful.

We have had an overwhelming amount of positivity towards this project and have a lot of meetings set up come July. These are focused on three areas: funding, advice and collaboration. The following subsections summarize the majority of companies we spoke to.

6.2.1 The Film Network

At the beginning of this project we discussed creating a tool for a new form of start up company possibly using Smart Contracts called the Distributed Autonomous Corporation (DAC). We were trying to find out how we could model the functions of a business in a trustless environment with the Blockchain.

The center for cryptocurrency research was approach by Gerard O'Malley, the CEO of The Film Network, a company created as part of the BBC which has since become independent. His idea was to create a Smart Contract for films, essentially comparing each film to a start up. This aligned perfectly with what we were looking for and in the background section it became clear that people we attempting similar ideas in the Music and Video Games industries too. This lead us to designing this project, a general solution applicable to all industries and demonstrating its use in the film industry.

We have had many meetings with Gerard and his partners throughout this project to ensure the ideas will be useful to film makers. Below is some of the feedback from these interactions:

1. End users do not want to have to concern themselves with Blockchain and Ethereum accounts. They would rather place some trust in us and create an account on a central server that takes care of the details under the hood. Users find having to run an Ethereum node locally to be a great barrier to entry.
2. Similar the the first point, users do not want to be aware of the different cryptocurrencies. So far we have only addressed the UK's independent film industry and they would like to pay and be paid in GBP and not have to exchange to Ether but I think it is safe to assume that most users would like to deal in their local fiat currency. Again in an ideal scenario, users would rather place some trust in us, and have our central server perform the exchange back and fourth on their behalf.

3. Originally we were opposed to adding a geographical location term to distribution contracts because it is difficult to trace the true origin of any web request. However, Gerard and his partners made clear to us how the Film industry is more traditional and we would be wiser to enable the current way of doing things even if it was not perfectly enforceable as having a transparent record of all transaction in the system would be enough for legal disputes even if it was not enforceable directly by the system.
4. Overall Gerard and partners were happy with the ideas and direction of the project which led to us applying for a grant from Innovate UK collaboratively discussed in the next subsection. They did mention that they would like it if we enabled companies like TFN to fully integrate the watch page of our system into theirs so that they could retain their users and give other companies an incentive to give their users bases access to our solution.

The first two points were unexpected to us coming from technical backgrounds. We had assumed users would not mind downloading an Ethereum client to use the new service if that meant not having to place much if any trust in us. It seems people are more willing to trust internet companies than we anticipated in favor of a better/simpler user experience.

For the third point we were also rather surprised. On the one hand we are trying to enable existing models of distribution and content management, however we are also trying to open up new possibilities and provide efficient solutions.

When we run our trial with the Film Network later this year we will be looking out for details such as adding geographical location of end users to distribution agreements, so that we may better address the existing methodologies. This research will have to be conducted for every industry we expand to, such as music, so that we may understand how they work in detail and thereby produce the best possible product.

The final point makes a lot of sense to us. Not only does it give companies an incentive to let their users access our solution it also means we do not have to create a separate site for each type of asset we host and deal with branding etc for the individual industries. It will also allow us to spread much faster tapping into existing user bases.

6.2.2 Innovate UK Application

As previously mentioned we applied for funding from Innovate UK's contest on the sharing economy. We were not successful and the feedback we received can be found in the second appendix. The challenge was to bring ideas from the sharing economy (e.g. Uber for sharing rides) to new industries and partner with Nesta, an innovation charity. We were looking for a £30,000 grant to further develop the system and integrate with TFN.

The assessment was conducted by 5 experts from various backgrounds who's feedback can be summarized as follows:

1. The project is described as "undoubtedly innovative", "compelling" and as having "substantial commercial novelty".
2. Insufficient detail about the commercialization of the project i.e. how much would we charge, how would users make money, how would it differ financially from other failed projects.
3. More work required on the testing plan i.e. why wasn't BAFTA named as a test partner, why wasn't more research completed on the specifics of the trial with NESTA.
4. The team seems to have all the necessary skills to turn this idea into a reality but weaknesses of the team have not been analyzed.

Given the positivity towards the project and that the weaknesses in our application were more centered around the structure of the collaboration and holes in the business side of things, specifically lacking financial projections, we will be applying for the next round which has a deadline in September for the collaborative aspects of this project.

6.2.3 British Academy of Film and Television Arts

Gerard had a connection to the British Academy of Film and Television Arts (BAFTA) which got us our initial meeting. We discussed the scope of the project in detail with 4 of BAFTAS employees including their research division and two programmers.

BAFTA were very interested in being a part of the project and were involved in many of the meetings with the film network mentioned above. The feedback listed in the film network section was echoed by BAFTA. They also have experience with applying for funding from Innovate UK, with something in the region of 8 funded projects. BAFTA provided help and guidance in the application to Innovate UK and will potentially be a part of our next application come September.

Going forward they have agreed to provide server space for hosting the films uploaded to our site if needed and will be providing advice on the film industry as well as potentially bringing us to film festivals to showcase our system.

6.2.4 Digital Catapult

We initially approached Digital Catapult with Gerard, and met with Sam Davies, the leader of a project they are developing in the video games industry as mentioned in the Background

research section 2.6.5. He was intrigued by the project and agreed to set up a meeting with everyone he knew currently developing Blockchain companies.

A few weeks later the cryptocurrency research center hosted an event at the Digital Catapult. We presented this project there to a large audience of industry professionals. The idea behind the event was to educate people about Blockchain and the new ways of solving problems it provides especially in government policy.

After the presentation we received a lot of positive feedback but also felt the idea had been slightly misunderstood by some. The term Smart Contract was one of the key problems to peoples understanding. We are not creating legal contracts that are on the Blockchain. We are creating a software application with code that runs on the Blockchain. Since the Blockchain is immutable and transparent an audit trail is available which is what provides the legal security. This concept is something we will need to make clearer in future presentations.

We spoke to the then CEO and CTO about the option of us working at the Digital Catapult but our interest did not align, however we are meeting them again come July. As a result of our discussions we will be looking at various options where we provide consultancy or a piece of equity in our company in return for advice. The new CEO used to be senior at the EMI music label and has a keen interest in Blockchain and so this project should be a good fit.

6.2.5 Government Round Table

The cryptocurrency research center hosted an event for the senior UK government members in an attempt to explain what Blockchain is and how promising it will be for the economy. We were allowed to briefly present this project at the event.

We were approached by various people after the presentation including ambassadors of British Embassies. All were very positive about the field and emphasized how important projects like this are to the UK's creative industries which are facing serious problems as described in the Background section 2.5.

The most astonishing development from this event was that Sir Mark Walport, the UK's chief science adviser saw and liked our presentation of this project and had his office reach out and request a copy of the slides. They also want to introduce us to Dr Ruth McKernan, the CEO of Innovate UK as they believe she would be interested in hearing more about it.

6.2.6 Ujo Music

We met Phil Barry, the founder of Ujo music, a company we had researched in section 2.6.4 at the digital catapult event. He is no longer working for Ujo but rather developing his own

company now trying to expand upon what was done at Ujo and bring it to the wider creative industries. He himself has a background in music production before doing an MBA at Oxford. We met with him a few weeks later to discuss the details of our projects and look at potential collaboration opportunities.

For the sounds of it, he is looking to do some similar aspects as we have done here. We discussed potentially founding a company together as he is still in the team building phase as well as other options such as his company potentially licensing our solution once completed.

Being in the industry himself he was overall very positive about the ideas, however given previous experience developing such Blockchain applications, he did advise us to do detailed testing of the costs involved as he found that in Ujo the because of the price of Ethereum gas the transactions did not end up being much cheaper than merely using normal banking services. More details on our analysis of this can be found in the next section.

6.2.7 Goldman Sachs

We met with David Reis, the head of technology investments for EMEA at Goldman Sachs. He told us that Goldman Sachs is currently doing a lot of research to learn about Blockchain and develop internal applications.

He liked the project but had limited experience in the creative industries. He advised us that there are currently no companies that are anywhere near mature enough for an institutional investor the size of Goldman. He does not expect any to be of adequate maturity for at least 3 or 4 years. We found this to be positive as it validated our research conclusions that we are one of the first in this space and that the industry is in its infancy.

6.3 Smart Contracts

In this section we will talk about some of the theoretical difficulties we faced using Ethereum and how these may be solved. We will also present our analysis of the 4 core components: Entity, Asset, Permission and Registry contracts.

6.3.1 Ethereum

Working with Ethereum was a lot more challenging than initially anticipated. There are two problems that have arisen out of the analysis of the Smart Contracts as a whole summarized below:

1. **Optimization for Gas:** When conducting user testing with our contracts deployed to a local Ethereum test network we experienced Out of Gas exceptions repeatedly. Each transaction in Ethereum that updates the state must compensate the network for the computation with gas. Gas is currently worth 20 GWei per unit. The maximum amount of gas that can be passed with any transaction is 3,000,000 which we always passed for the purposes of the test which equates to 0.06ETH, approximately 0.60 GBP at the time of writing. During the analysis we found that contracts creating other contracts was much more expensive than if we created the two contracts ourselves externally and then just validated what we had created in Ethereum. This has led to a complete redesign of the system since in an effort to be as concise as possible and limit inter-contract communication to constant method calls rather than state updating ones.
2. **Blockchain Split:** In any Blockchain based system, the longest chain is considered the correct one. So if two nodes on opposite sides of the network mine different blocks at the same time the Blockchain splits and is eventually deterministic once one of the branches becomes longer than the other. This behavior means that once we receive an address for a contract, it could change if the Blockchain has split. This became a problem for multiple aspects of our site. We cannot guarantee the order in which the transactions into our site will be processed and we cannot be sure of the return value of transactions until waiting for a few blocks to be confirmed after the block containing the transactions. The first problem is solvable by batching transactions. The idea is simple create a batch of transactions with an ordering on them and submit the batch to the network. The second issue is a lot more complicated. The private server can observe the Blockchain and make the required amendments to its local storage. However the user interface needs to be extremely reactive, constantly updating the addresses displayed on the site to avoid us entering any batches of transactions into Ethereum which reference old incorrect values. This may be unavoidable at times and needs further research to ensure consistency.

6.3.2 Contract Optimization

Here we present the analysis of our contract optimization both in terms of design changes for efficiency and reworking algorithms to be more computationally efficient figure 6.1. The final data point for each contract is as a result of using the Solidity compilers advanced optimization mode.

It should be noted that we did not find ourselves with enough time to implement decentralized decision making for entities. This is mainly because decisions become rather

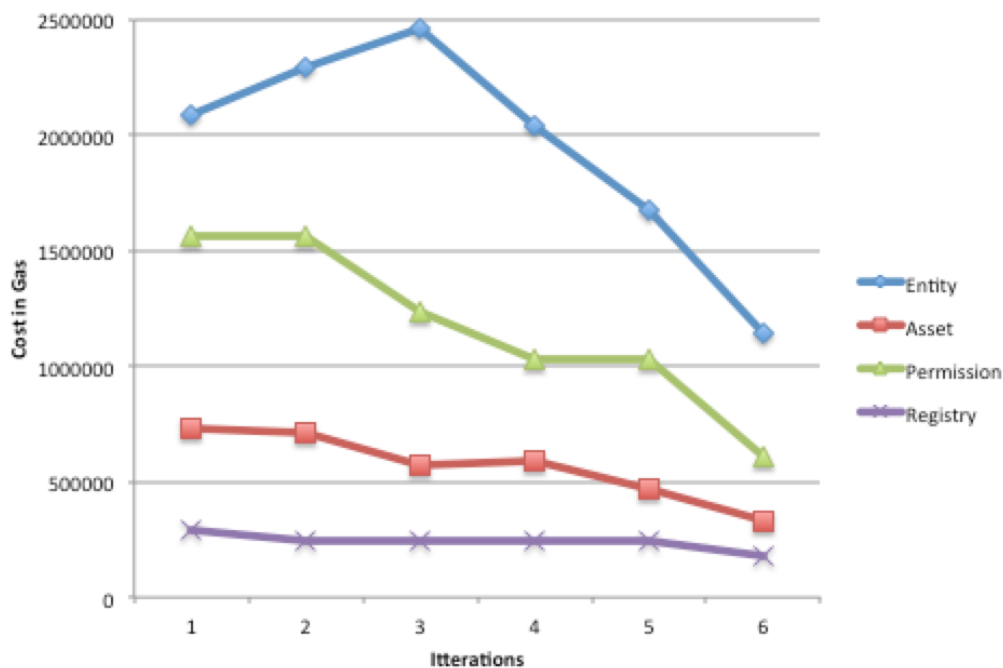


Fig. 6.1 Optimization of Smart Contracts.

complicated in our hierarchy of individuals and entities and described in detail in sections 3.3.2 and 5.2.3. Whilst this is not essential to the core of our project it will be the first aspect we work on next.

We also were not able to add the negotiation process for distribution contracts described in section 3.2.2. The cause of this is that it takes too much storage to save each of the propositions in Ethereum contracts. We will solve this problem by maintaining a hash of the propositions stored on IPFS in the Ethereum contract. Upon signing of a permission, users will pass along the hash of the proposition they agree with.

6.4 Private Server

In this section we will provide a summary of the analysis we performed on the private server. It is important to state from the outset that we feel we have performed weakly in testing this component of our system. We did not have the time we would have liked to do the scope of testing required to prove security.

6.4.1 Asset Storage

As mentioned in sections 5.4.3 and 3.4 we used IPFS to store our assets. Having analyzed this decision we came to the conclusion that this is actually rather pointless. Our private server encrypts the asset uploaded to it and saves it to the local IPFS node. When a user requests to view the asset they do not access it directly on IPFS but rather contact the server which accesses it in its local node. This means the asset will never be stored on any of the other peers and thus we will not benefit from the IPFS network at all as shown in figure 6.2. All we have essentially done is create a local database where everyone on IPFS can access our encrypted assets if they so wished, thereby reducing the security provided by a central server and still having the same disadvantages of a single point of failure.

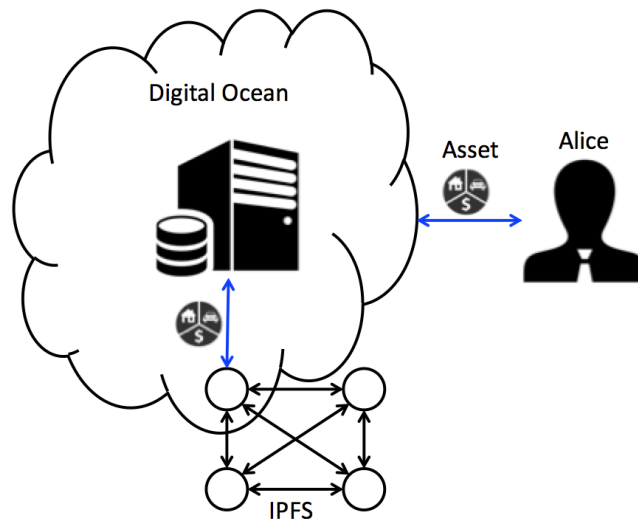


Fig. 6.2 Server only communicating with local IPFS node. [85] [31] [49]

We have come to the conclusion that it would make much more sense to use StorJ or ideally Maidsafe when it launches, so that the files are split up and distributed throughout the network. Offering the remaining storage space on the private server to the SAFE network would help mitigate the cost of using it. This will be one of the first aspects we address when further developing the application.

6.4.2 Threat Analysis

Our system is vulnerable to collision attacks. Collision attacks are possible because hash functions return a constant output size. This means if the input domain to the hashing function is larger than the range some inputs will map to the same hashes. Since hashes are involved in creating users' addresses, if someone were to find a collision in the hash function, they

could perform sensitive operations as another address without possessing the password to it. The biggest implications this would have for our site is if a user found a collision for an address which owns an Entity. In this case they would be able to transfer out the balance to an account of their choosing. However, since Ethereum has had no problems this far it is unlikely this will ever be a problem. We are no less secure than Ethereum in this respect. Also this attack is very hard to have success with. Even if an attacker found a collision, there is tiny chance that either of the colliding inputs was used for any of the accounts on our site.

6.4.3 Security and Testing

We used the latest standards for encryption of our database and the assets as shown in section 5.4. In future we would like to post an add inviting people to attempt hacking the server. We will also make the source code of the application public since all keys for encryption are stored in environment variables on the server itself for security. We will approach industry security experts and ask for advice whilst running them through our source code.

We did not have sufficient time to adequately test the application. We performed integration tests by querying the API with various parameters and validating the return values, however we should have performed unit testing on the most critical components for security such as the encryption of the assets and the encryption of the database.

6.5 Decentralized Application (DApp)

6.5.1 User Experience Feedback

Since the demographics we identified as potential users of the web application in general do not have technical backgrounds, we devoted a significant amount of time to designing a user interface that would be intuitive for a non-technical individual to navigate, and that would explain all of the Blockchain-related concepts to the user without adding too much clutter and confusion. To achieve this objective, we organized 2 rounds of user feedback at different points in the development:

1. A live interview where individuals clicked-through an *InVision* prototype of the initial mock-ups (online at).
2. A SurveyMonkey questionnaire sent to a selection of potential users along with a link to the working prototype (with the main permission, entity, and asset functionality in place) (online at).

The first live interview at the mock-up stage was performed on 4 potential users: 1 film maker, 2 potential film distributors (1 Youtube Vlogger and 1 university student studying marketing), and 1 film consumer. The interviewees each spent around 5 minutes clicking through the design prototype unassisted, commencing at the entry point of their relevant user categories (e.g. the film maker started at the landing page). We gained the following insights:

- All candidates (100%) said they were confused by the Ethereum addresses for the assets, entities, and permissions. The main reason for this was that their own Ethereum addresses were used to create entities, which then create permissions and assets, so they consistently confused these addresses with their own. Our response to this was to show the name whenever possible, e.g. in the *view entity* page in the Appendix, we replaced the asset addresses in the asset and permission thumbnails with their actual names, as demonstrated below in figure 6.3.

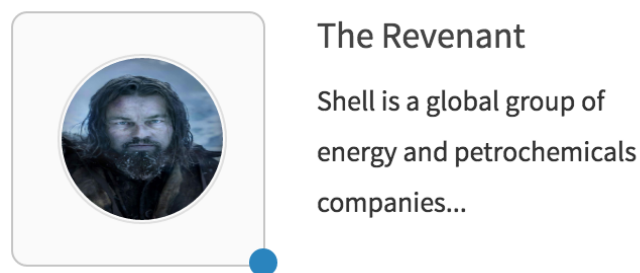


Fig. 6.3 The new asset thumbnail on the view entity page designed after the first round of user feedback.

- Furthermore, all candidates (100%) also suggested removing the "active address" header on the dashboard/registry page (see the Appendix), since it was redundant because of the active address displayed in the navigation bar.
- 3 candidates (75%) said they were confused by the term "permission". When asked what a permission was, candidates including one in the "film distributor" user category did not understand that this was analogous to a distribution contract. Hence, throughout the user interface we replaced all references to "permission" with "distribution contract".
- 2 candidates (50%) said they were confused by the term "entity". They suggested adding an information section somewhere on the dashboard/registry page to explain what it was. We decided to add an on-hover popover with an explanation about what an entity is.

- 2 candidates (50%) suggested listing the parent and child entities in the permission thumbnail on the view entity page, since it is important to know who the owners of an entity are in business with. We agreed with this suggestion, and changed the thumbnails to match that shown in figure 6.4.

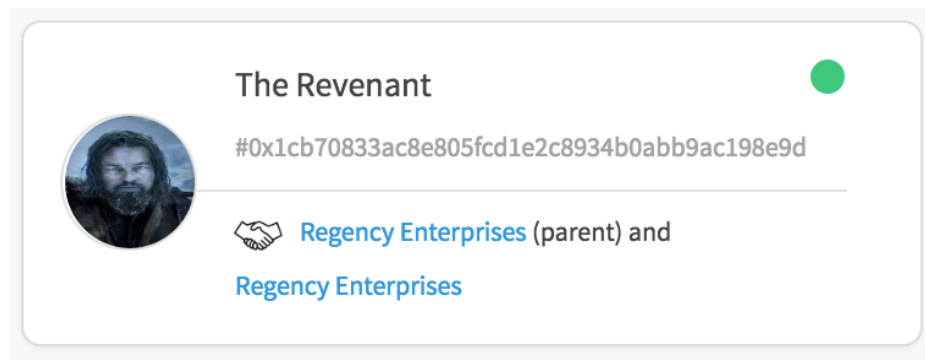


Fig. 6.4 The new permission thumbnail on the view entity page designed after the first round of user feedback.

- 1 candidate (25%) said they were confused by the term "asset", but when asked what an asset was, they understood that this was analogous to "film". We chose not to change references to "asset" with "film" for now, but we will before a beta launch.
- Finally, all candidates (100%) were able to explain how the site worked vaguely. Although there was some confusion about whether the distribution occurred in a tree-like format or whether every distributor interacted with the asset owner, they all found the overall navigation through the user interface intuitive.

The final round of user-feedback was done after the final version of the product was produced, and after the changes to the user interface from the first round of feedback were implemented. The idea here was to understand the strengths and weaknesses of the finalised user interface to learn what changes need to be ultimately made before launching the product.

A SurveyMonkey survey was created, along with a live Heroku version of the prototype. We had 21 total responses, 4 (19%) of which were conducted in person (in a live interview setting), the remaining 17 online.

Respondents were instructed to first determine which user category they fit into: film maker, film distributor, or film consumer, and based on this visit /MAKER, /DISTRIBUTOR, /CONSUMER, and navigate through the website from there. 10 out of 21 (47.6%) said they were film consumers, 6 out of 21 (28.6%) said they would be film distributors, and 5 out of 21 (23.8%) said they would be film makers, as demonstrated in figure 6.5. Further, 17 (81%) of respondents said they were from non-technical backgrounds.

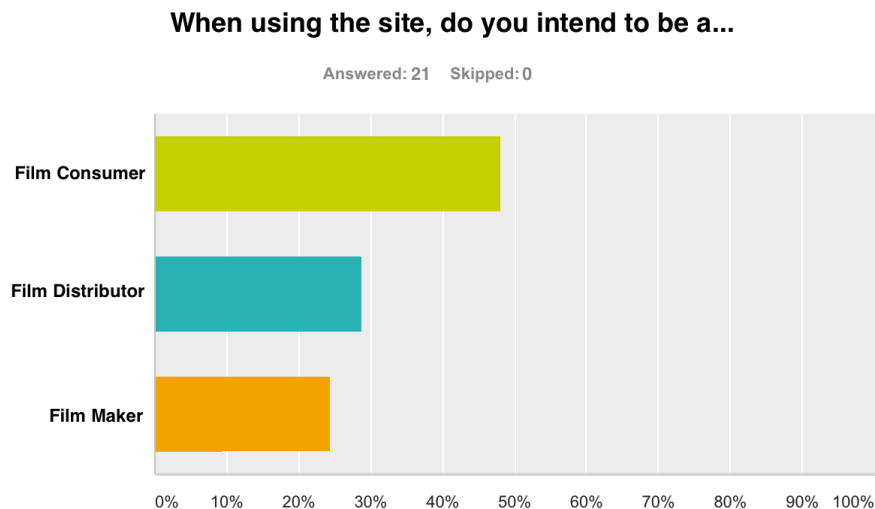


Fig. 6.5 The survey respondent breakdown into the 3 user groups.

They were then asked a series of questions that tested understanding of some of the concepts (what an entity, asset, and distribution contract are), and asked what they did once they reached a page (along with if, on a given page, there was anything they did not understand or anything they suggested changing). Some of the questions generating interesting responses were:

- As shown in figure 6.6, we asked users what they thought the application was for after reading the landing page. 16 users (76.2%) selected the correct answer (answer 2), "it uses Blockchain technology to give you a new distribution channel for your film, where anyone can become a distributor, sell views, and split the revenue per sale with the film maker/parent distributors". However, 4 users (19%) said "it uses Blockchain technology to allow you to upload your favorite film to the website and allow you to make money by selling views of it" (answer 1), a wrong answer. Since this answer would be accurate if "favorite film" were replaced with "your film", we estimate a fraction of these incorrect responses (2 out of 4) were due to misreading, since their responses to the other questions demonstrated that they understood how the site worked. Interestingly, all 4 of these incorrect responses were produced by users defining themselves as film consumers. Finally, 1 user (4.8%) selected the other wrong answer (answer 3), "it uses Blockchain technology to permit any person from anywhere to buy your film". This user along with the 2 previously discussed (a total 14.2% of users) missed the fundamental idea of what the web application does, elucidating a weakness in our technical explanations to non-technical individuals.

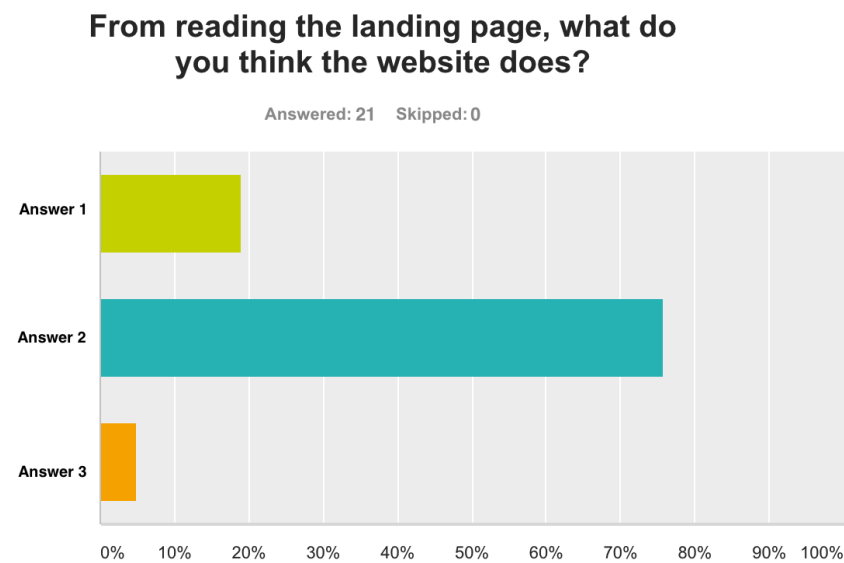


Fig. 6.6 The survey responses to the question: "From reading the landing page, what do you think the website does?".

For this reason, we have decided to add a panel at the bottom explaining in detail how it works, with specific emphasis on mentioning the terms "entity", "asset", and "distribution contract", along with the image of a distribution tree as depicted below in figure 6.7.

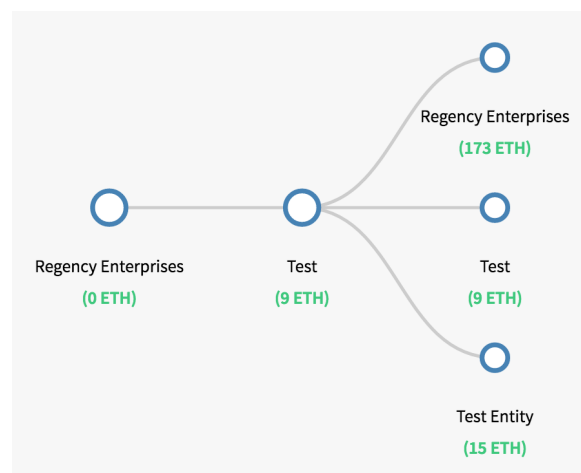


Fig. 6.7 A screen shot of a distribution tree added to the landing page.

This will be added to the landing page before production.

- Since the main changes to the interface after the first round of user feedback concerned providing more information and help content with respect to explaining terms such

as "entity" and "asset", we wanted to see if users' understanding had improved. We therefore asked the question "did you find the terms entity, asset, and distribution contract confusing?", as depicted in figure 6.8. 9 users (42.8%) said that the terms were clear, and 7 users (33.3%) said the terms became clear after reading through the help information. Thus, the overall comprehension of the basic application concepts increased from 50% to 76.1%, implying that the changes were successful. Unfortunately, 5 users (23.9%) still had trouble understanding what these terms were, and in a follow-up question asking if they had read the help content, 3 of these users said they had not seen any extra information. Hence, before launch, we must make the help content and other information easier to find.

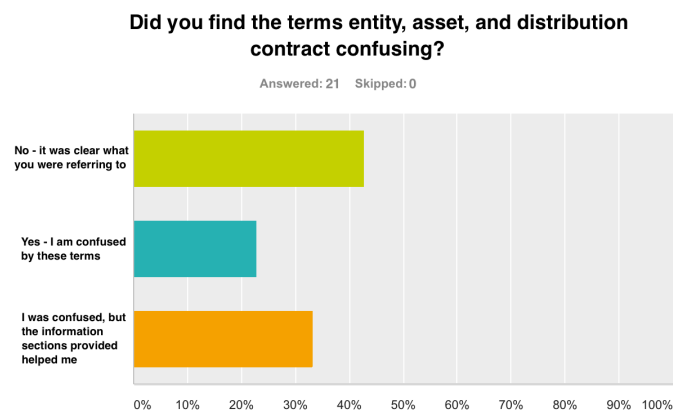


Fig. 6.8 The survey responses to the question "did you find the terms entity, asset, and distribution contract confusing?".

- All users (100%) said that they found the user interface to be intuitive and easy to use and navigate: comments left by respondents included mentioning that button names and actions on each page were clear, and that the information displayed on each page was easy to understand and big enough.

From the observations of the second round of user feedback, we can identify a set of strengths and weaknesses with respect to the user interface. Overall, the site was found to be easy to navigate and respondents said that the interface was clean and intuitive. Hence, a strength is that our non-technical user base is able to navigate the site without issues. The main weakness identified is that users' conceptual understanding of how the website works and its purpose could be significantly improved. A fraction of users struggled to find the help content, and other users simply did not grasp what the website was for.

6.5.2 Testing

A note should be made on our choice of testing frameworks. We used Jasmine for the Smart Contracts and Mocha/Chai for the decentralized application itself. This meant we had to learn two different systems and configure how to run both testing suits automatically in conjunction. This proved to be a bad decision as it caused a lot more difficulty than if we had just chosen one to work with. Going forward we will be testing all aspects of the Smart Contracts and the Front End Application using Jasmine.

6.6 Strengths and Weaknesses

This section will provide an overview of what we have found by evaluating this project. We will list strengths and weaknesses in terms of the Background research where applicable.

Strengths:

- Our application is the first to enable the secure distribution of digital assets and management of the associated permissions, without a central party. Users' distribution agreements provide a transparent audit trail on the Blockchain.
- Despite being completely transparent, our site protects the identity of the individual users consuming assets. There is no need to sign up or provide any personal information. Users can buy and stream videos with complete anonymity.
- With a private server handling all sensitive encryption keys for the assets and by storing the encrypted assets on a decentralized system, film creators can rest assured that their content is safe and cannot be accessed by anyone without the permission to do so.
- By managing payments on the Ethereum network we can provide a much lower cost per transaction, meaning higher profit margins for distributors and film makers. It also means we can send micro payments and trickle every individual purchase from a user up the distribution chain into the Entities' accounts preventing any short term cash flow issues.
- Having chosen to write all Contracts in Solidity we can not only deploy our contracts on Ethereum, but also deploy them on the new side chain RootStock, which is bringing Smart Contracts to Bitcoin. This means we are not dependent on the success of any individual Smart Contract platform but rather on the the success of the Blockchain field.

Weaknesses:

- The speed of the system when it comes to any action which requires the creation of an Ethereum transaction is inherently limited by the block confirmation time which is approximately 15 seconds. This can negatively impact users in a number of ways for example: If a user were to buy a view for a film and attempt to stream it less than 15 seconds later, they would not be permitted to do so.
- In the Ethereum model users are anonymous, merely identified by their public keys. This results in a potential mechanism to abuse our site for criminal activity as currently implemented. We have no means of validating that a person uploading and claiming they own some asset actually does. Consider a pirated video online. A user would be able to illegally download a movie, upload it to our site and start making money by selling it.
- In extreme cases when the Blockchain splits into n branches and in that moment a user inserts some transaction into the Blockchain referencing an address of a contract affected by the split, that transaction has a $1/n\%$ chance of referencing the valid contract. In practice we could never force this to happen but it is theoretically possible.
- We have not performed sufficient security testing on our private server to ensure there is no known way of attacking it. However, even if it is successfully hacked, all database information is encrypted so it is debatable as to how severe a problem this is at this stage. This is the first item we intend to address when further developing this product.

Chapter 7

Conclusion

7.1 Lessons Learned

We have come across many interesting problems throughout this project. This has given us a great insight into how powerful Smart Contracts are, the potential that they have and the challenges they are currently facing. The idea of a decentralized internet is exciting, removing the need to trust third parties and giving us many options that were previously just not possible. However it is very early days in this field and it remains to be seen to what extent it may be regulated and how we can truly guarantee security going forward with computational power in the world always increasing and therefore vulnerabilities being found in cryptographic principals.

In addition, we have become aware of what an impact this project could have on the creative industries in particular. Problems such as rights attribution and distribution of digital content have long been prevalent. It is clear that the next years will dramatically improve such industries. Furthermore it has become clear how topical Blockchain companies are at the moment and much funding is available for start ups attempting to build solutions using these new tools. For our perspective we will be attempting to found a company and further develop this product starting in the Film and Music Industries.

Unsurprisingly we have discovered a distrust from the general public when it comes to Blockchain. As with any new technology users need time to become accustomed to it. Many capable industry professions struggle to understand how Blockchain works fundamentally and how it provides security. This has shown us how important it is to obscure any of the technical details of the Blockchain from our users during this "warm up" phase.

Surprisingly, we have found that there are many businesses looking to hire us off the back of this project. This is true mainly for the financial sector or individuals with business experience looking for a more technologically orientated founder or Chief Technology Officer

for their start up. The creative sectors have been very willing to collaborate and build systems together. This has given us even more motivation to keep developing and improving the system in an attempt to build a start up ourselves.

Finally we have been able to observe how quickly the Blockchain industry as a whole is evolving. Whilst working on the project much has changed and the rapid progression of the field is gaining momentum. During the course of this project we have had to rework aspects given new opportunities that presented themselves. We have attempted to list all the new systems, too immature for this project at the time of writing, but worth keeping an eye on to better aspects of our work in the future.

7.2 Future Work

We could go on for many pages listing all of the extensions and improvements we would like to add to this project, however we will summarize those that we find the most interesting:

- The first extension we will add to the project is by adding more functionality to the server API. We will allow users to make the typical accounts they are used to from today's web applications. Under the hood we will automatically create an Ethereum address for them and associate it with their user account. If the user needs more Ether they will be able to send us the fiat currency and under the hood we will exchange it and add it to their account. This will mean the user can be completely oblivious to Ethereum. The disadvantage is that they will have to trust us but this does not seem to be a problem in today's model. We will still allow users with local Ethereum nodes to use the site as before and will slowly start converting the traditional account holders to the Ethereum way.
- Once the Enigma project being built at MIT launches we will attempt to remove the need for the server handling the sensitive information. They claim to have a solution to privacy in the Blockchain that requires no trust between any of the parties and has an innovative solution using optimized versions of algorithms from Secure Multi-Party Computation and Secret Sharing Schemes. This would allow us to create an application that requires no trust what soever.
- Once Mailsafe launches an application we will store all Assets on the SAFE network. This means will require a lot less storage on the server but will have to pay the SAFE network for the resources consumed. It will however provide better security because the file will be split over many different nodes.

- We will expand the the music industry next and provide basically the same functionality and from there options are the games industry, photography and any other digital assets such as documents. The system has been designed for this and will only require a new watch/view page for each type of asset.
- We will expand the information that can be stored on each Smart Contract, by allowing users to add images and detailed descriptions of their assets. Ethereum is not suited for this and so we will use IPFS to store the content and save the hash IPFS address in the corresponding Ethereum contract.

These some of the largest pieces of future work we have identified. If you have any suggestions or would like to be a part of further building this platform please get in touch with us.

References

- [1] Allison, I. (2015). UBS reveals its interest in Sidechains as well as Ethereum. <http://www.ibtimes.co.uk/ubs-reveals-its-interest-sidechains-well-ethereum-1519706>. Online; Accessed 26 Jan. 2016.
- [2] andrewmagdurulan.files.wordpress.com (2016). Figure of P2P Network. <https://andrewmagdurulan.files.wordpress.com/2012/11/peer-to-peer-network.jpg>. Figure; Accessed 08 Jun. 2016.
- [3] Apache Software Foundation (2015). SSL/TLS Strong Encryption: An Introduction - Apache HTTP Server Version 2.2. http://httpd.apache.org/docs/2.2/ssl/ssl_intro.html#cryptographictech. Online; Accessed 26 Jan. 2016.
- [4] Assistant US Attorney (2015). Sealed Complaint. <https://www.cs.columbia.edu/~smb/UlbrichtCriminalComplaint.pdf>. Online; Accessed 17 Apr. 2016.
- [5] BAFTA (2016). BAFTA's logo. http://www.bafta.org/sites/all/themes/bafta_theme/images/logo_master.png. Figure; Accessed 08 Jun. 2016.
- [6] Becker, G. (2008). Merkle Signature Schemes, Merkle Trees and Their Cryptanalysis. http://www.emsec.rub.de/media/crypto/attachments/files/2011/04/becker_1.pdf. Online; Accessed 26 Jan. 2016.
- [7] Benet, J. (2016). IPFS - Content Addressed, Versioned, P2P File System. <https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf>. Online; Accessed 13 Jun. 2016.
- [8] Betcher R. (2016). Figure of hash function. <http://image.slidesharecdn.com/securehashingalgorithm-121216002452-phpapp01/95/secure-hashing-algorithm-6-638.jpg?cb=1355618469>. Figure; Accessed 26 Jan. 2016.
- [9] Bitcoin Foundation (2016a). Figure of Bitcoin Proof of Work. <http://www.slideshare.net/tsasaa12/bitcoin-cryptocurrency-48132020>. Figure; Accessed 17 Apr. 2016.
- [10] Bitcoin Foundation (2016b). Figure of Bitcoin Transaction Chain. http://4.bp.blogspot.com/-SHP3b9uqD70/VAuEY5WdHLI/AAAAAAAAAFzw/UEcB0fkVSMU/s1600/bitcoin_transaction_ownership_chain.png. Figure; Accessed 17 Apr. 2016.
- [11] Bitcoin Foundation (2016c). Figure of Bitcoin Transaction Signing. http://www.4flush.com/wp-content/uploads/2013/03/Bitcoin_Transaction_Visual.png. Figure; Accessed 17 Apr. 2016.

- [12] Bitcoin.org (2016). Developer Guide - Bitcoin. <https://bitcoin.org/en/developer-guide#p2pkh-script-validation>. Online; Accessed 26 Jan. 2016.
- [13] Bittunes (2016a). Bittunes About Page. <http://bittunes.co.uk/about-bittunes/>. Online; Accessed 23 Apr. 2016.
- [14] Bittunes (2016b). Bittunes Home Page. <http://www.bittunes.org>. Online; Accessed 23 Apr. 2016.
- [15] British Film Institute (2001). 7 Challenges Facing Independent Filmmakers. <http://www.bfi.org.uk/sites/bfi.org.uk/files/downloads/uk-film-council-a-filmmakers-guide-to-distribution-and-exhibition-2001.pdf>. Online; Accessed 26 Jan. 2016.
- [16] British Film Institute (2015). In Hollywood, Sony Hack's Chilling Effect On Movie Pipeline Is Already Being Felt. <http://www.bfi.org.uk/sites/bfi.org.uk/files/downloads/bfi-uk-film-market-as-a-whole-2015.pdf>. Online; Accessed 26 Jan. 2016.
- [17] Buterin, V. (2014a). Ethereum and Oracles - Ethereum Blog. <https://blog.ethereum.org/2014/07/22/ethereum-and-oracles/>. Online; Accessed 26 Jan. 2016.
- [18] Buterin, V. (2014b). Merkle Tree Image. <https://chrispaciac.files.wordpress.com/2013/09/merkle-tree.jpg>. Online; Accessed 26 Jan. 2016.
- [19] Buterin, V. (2014c). Side Chains: The How, The Challenges and the Potential. <https://bitcoinmagazine.com/articles/side-chains-challenges-potential-1397614121>. Online; Accessed 26 Jan. 2016.
- [20] Buterin, V. (2014d). Toward a 12-second Block Time. <https://blog.ethereum.org/2014/07/11/toward-a-12-second-block-time/>. Online; Accessed 26 Jan. 2016.
- [21] Buterin, V. (2015). Ethereum Wiki on Github. <https://github.com/ethereum/wiki/wiki/Problems>. Online; Accessed 26 Jan. 2016.
- [22] Cassano, J. (2014). What Are Smart Contracts? Cryptocurrency's Killer App. <http://www.fastcolabs.com/3035723/app-economy/smart-contracts-could-be-cryptocurrencys-killer-app#>. Online; Accessed 26 Jan. 2016.
- [23] Cisco (2016). Cisco Figure of Internet Traffic. <http://www.advancedeidetic.net/wp-content/uploads/2014/09/CiscoVideoGrowth.jpg>. Figure; Accessed 17 Apr. 2016.
- [24] cloudfront.net (2016a). Icon of Ballet. <https://d30y9cdsu7xlg0.cloudfront.net/png/14292-200.png>. Figure; Accessed 09 Jun. 2016.
- [25] cloudfront.net (2016b). Icon of contract. <https://d30y9cdsu7xlg0.cloudfront.net/png/42571-200.png>. Figure; Accessed 09 Jun. 2016.
- [26] Codius.org (2016a). Architecture - Codius Docs. <https://codius.org/docs/overview/architecture>. Online; Accessed 26 Jan. 2016.
- [27] Codius.org (2016b). What is Codius? - Codius Docs. <https://codius.org/docs/overview/what-is-codius>. Online; Accessed 26 Jan. 2016.

- [28] Coinmarketcap (2016). Figure of Bitcoin Market Cap. <https://coinmarketcap.com/currencies/bitcoin/#charts>. Figure; Accessed 17 Apr. 2016.
- [29] Counterparty.io (2016a). About | Counterparty. <http://counterparty.io/platform/>. Online; Accessed 26 Jan. 2016.
- [30] Counterparty.io (2016b). Counterparty Recreates Ethereum's Smart Contract Platform on Bitcoin. <http://counterparty.io/news/counterparty-recreates-ethereums-smart-contract-platform-on-bitcoin/>. Online; Accessed 26 Jan. 2016.
- [31] dallastxdivorce.com (2016). Icon of Asset. <http://www.dallastxdivorce.com/wp-content/uploads/sites/403/2015/10/icon-division-marital-property.png>. Figure; Accessed 09 Jun. 2016.
- [32] Das S. (2016). Euro Parliament to Host Blockchain and Virtual Currencies Crash Course. <https://www.cryptocoinsnews.com/euro-parliament-host-blockchain-virtual-currencies-crash-course/>. Online; Accessed 17 Apr. 2016.
- [33] Doz.com (2016). Figure of Viral spreading of content. <http://www.doz.com/cms/wp-content/uploads/2014/06/viral.png>. Figure; Accessed 08 Jun. 2016.
- [34] EET.com (2016). Figure of Elliptic Curve. http://m.eet.com/media/1170221/point_addition.png. Figure; Accessed 26 Jan. 2016.
- [35] Ethereum Foundation (2016). Figure of Ethereum Node Decision Process. <http://images.rapgenius.com/e842e23278d4c69ef6968cc826b20c5f.679x1000x1.png>. Figure; Accessed 17 Apr. 2016.
- [36] Ethereum.gitbooks.io (2016a). Gas and transaction costs. <https://ethereum.gitbooks.io/frontier-guide/content/costs.html>. Online; Accessed 26 Jan. 2016.
- [37] Ethereum.gitbooks.io (2016b). Introduction | Ethereum Frontier Guide. <https://ethereum.gitbooks.io/frontier-guide/content/mining.html>. Online; Accessed 26 Jan. 2016.
- [38] Etherscan.io (2016). Ethereum Gas Price Chart. <https://etherscan.io/charts/gasprice>. Online; Accessed 26 Jan. 2016.
- [39] Filmfund (2016). Film Fund Home Page. <http://www.filmfund.io/#!/how-it-works/c1xdc>. Online; Accessed 23 Apr. 2016.
- [40] Florincoin (2016). Homepage. <http://florincoin.org/>. Online; Accessed 21 Apr. 2016.
- [41] GitHub (2014). Codius Repository. <https://github.com/codius/codius>. Online; Accessed 26 Jan. 2016.
- [42] Hayase, N. (2015). The Blockchain is a New Model of Governance. <http://www.coindesk.com/consensus-algorithm-and-a-new-model-of-governance/>. Online; Accessed 26 Jan. 2016.
- [43] Huth M. (2016a). Slides for CO409 Cryptography Engineering. Lecture Notes; Accessed 26 Jan. 2016.

- [44] Huth M. (2016b). Slides for CO409 Cryptography Engineering. Lecture Notes; Accessed 26 Jan. 2016.
- [45] iconfinder.com (2016a). Icon of Distributor. <https://cdn4.iconfinder.com/data/icons/professionals/512/distributor-512.png>. Figure; Accessed 09 Jun. 2016.
- [46] iconfinder.com (2016b). Icon of Home Owner. <https://cdn2.iconfinder.com/data/icons/people-icons-3/72/04-512.png>. Figure; Accessed 09 Jun. 2016.
- [47] image.freepik.com (2016a). Figure of Hash icon. https://image.freepik.com/free-icon/hash_318-10163.jpg. Figure; Accessed 11 Jun. 2016.
- [48] image.freepik.com (2016b). Figure of HTML document. https://image.freepik.com/free-icon/html-file-with-code-symbol_318-45756.png. Figure; Accessed 11 Jun. 2016.
- [49] image.freepik.com (2016c). Figure of Server icon. https://image.freepik.com/free-icon/server_318-46695.png. Figure; Accessed 11 Jun. 2016.
- [50] Image.slidesharecdn.com (2016a). Ethereum Whitepaper - Blockchain. <http://image.slidesharecdn.com/stateofethereumandmining-140630121009-phpapp02/95/state-of-ethereum-and-mining-7-638.jpg?cb=1404130347>. Online; Accessed 26 Jan. 2016.
- [51] Image.slidesharecdn.com (2016b). Ethereum Whitepaper - State Transition Function. <http://image.slidesharecdn.com/ethereum-whitepaper-140619054212-phpapp02/95/ethereum-whitepaper-17-638.jpg?cb=1403417440>. Online; Accessed 26 Jan. 2016.
- [52] Incapsula.com (2016). Denial of Service Attacks. <https://www.incapsula.com/ddos/ddos-attacks/denial-of-service.html>. Online; Accessed 26 Jan. 2016.
- [53] International Trade Administration (2016). 2015 Top Markets Report Media and Entertainment. http://trade.gov/topmarkets/pdf/Media_and_Entertainment_Top_Markets_Report.pdf. Online; Accessed 17 Apr. 2016.
- [54] Jackson D. (2016). Ethereum Self-Crowdfunded \$21 Million in Less than a Month — What Is It and Why Should We Care? <http://insidebitcoins.com/news/ethereum-self-crowdfunded-21-million-in-less-than-a-month-what-is-it-and-why-should-we-care/23838>. Online; Accessed 17 Apr. 2016.
- [55] Krug, J. (2015). Why Ethereum? <http://www.augur.net/blog/why-ethereum>. Online; Accessed 26 Jan. 2016.
- [56] Kukarkin, A. (2016). Mediachain: a distributed metadata protocol. <https://blog.mediachain.io/mediachain-483f49cbe37a#.7coc6m8ea>. Online; Accessed 10 Jun. 2016.
- [57] Lee B. (2016). Netflix chief: Beasts of No Nation has already had over 3m views. <http://www.theguardian.com/film/2015/oct/27/beasts-of-no-nation-netflix-three-million-views-original-feature>. Online; Accessed 17 Apr. 2016.
- [58] Lerner, S. (2015). RSK - Bitcoin Powered Smart Contracts. <https://uploads.strikinglycdn.com/files/90847694-70f0-4668-ba7f-dd0c6b0b00a1/RootstockWhitePaperV9-Overview.pdf>. Online; Accessed 26 Jan. 2016.

- [59] Lodderhose, D. (2014). Movie piracy: threat to the future of films intensifies. <http://www.theguardian.com/film/2014/jul/17/digital-piracy-film-online-counterfeit-dvds>. Online; Accessed 26 Jan. 2016.
- [60] Lovingood Studios (2015). Bitcoin 2.0 Meets Hollywood: Filmfund.io Launches TIX crowdsale With Oscar-Winning Team Hoping to Raise 3,000 BTC. <http://www.coindesk.com/press-releases/bitcoin-2-0-meets-hollywood-filmfund-io-launches-tix-crowdsale-with-oscar-winning-team-hoping-to-raise-3000-btc/>. Online; Accessed 23 Apr. 2016.
- [61] Maidsafe (2016a). Features of the SAFE network. <http://maidsafe.net/features.html>. Online; Accessed 18 Apr. 2016.
- [62] Maidsafe (2016b). Maidsafe Proof of Work Figure. <http://maidsafe.net/safecoin.html>. Figure; Accessed 18 Apr. 2016.
- [63] Maidsafe (2016c). The Architecture of the SAFE Network. <https://blog.maidSAFE.net/tag/internet-protocol/>. Online; Accessed 18 Apr. 2016.
- [64] Maxim, J. (2015). Ripple Discontinues Smart Contract Platform Codius, Citing Small Market. <https://bitcoinmagazine.com/articles/ripple-discontinues-smart-contract-platform-codius-citing-small-market-1435182153>. Online; Accessed 26 Jan. 2016.
- [65] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. <https://bitcoin.org/bitcoin.pdf>. Online; Accessed 26 Jan. 2016.
- [66] Parker, L. (2015). Rootstock Is Coming, Are Ethereum's Days Numbered, Or Will The \$18 Million Dollar Idea Survive. <http://bravenewcoin.com/news/rootstock-is-coming-are-ethereums-days-numbered-or-will-the-18-million-dollar-idea-survive/>. Online; Accessed 26 Jan. 2016.
- [67] Petzold, C. (2008). The annotated Turing. Book.
- [68] Pomerantz, D. (2014). In Hollywood, Sony Hack's Chilling Effect On Movie Pipeline Is Already Being Felt. <http://www.forbes.com/sites/dorothyPomerantz/2014/12/18/sony-hack-could-have-a-chilling-effect-on-hollywood/#58b752b63072>. Online; Accessed 26 Jan. 2016.
- [69] Prableen Bajpai, C. (2016). Blockchain Definition | Investopedia. <http://www.investopedia.com/terms/b/blockchain.asp>. Online; Accessed 26 Jan. 2016.
- [70] Rizzo P. (2016). World Economic Forum Gears Up for Hands-On Blockchain Research in 2016. <http://www.coindesk.com/world-economic-forum-blockchain-research-2016/>. Online; Accessed 17 Apr. 2016.
- [71] Schadelbauer R. (2016). Informa: Global OTT Video Market to Grow to \$37B by 2017. <http://www.ntca.org/new-edge/video/informa-global-ott-video-market-to-grow-to-37b-by-2017>. Online; Accessed 17 Apr. 2016.

- [72] Sia Tech (2016). Roadmap. <http://sia.tech/about/roadmap>. Online; Accessed 17 Apr. 2016.
- [73] Sompolinsky, Y. and Zohar, A. (2016). Accelerating Bitcoin's Transaction Processing Fast Money Grows on Trees, Not Chains. http://www.cs.huji.ac.il/%7Eavivz/pubs/13/btc_scalability_full.pdf. Online; Accessed 26 Jan. 2016.
- [74] Stafford, P. (2015). The blockchain and financial markets. <http://www.ft.com/cms/s/0/454be1c8-2577-11e5-9c4e-a775d2b173ca.html#axzz3yLrFVwXR>. Online; Accessed 26 Jan. 2016.
- [75] Storj Company (2016). Storj Labs Launches Beta. <http://blog.storj.io/>. Online; Accessed 21 Apr. 2016.
- [76] Storj.io (2016). Figure of Storj File Distribution. <https://storj.io/>. Figure; Accessed 08 Jun. 2016.
- [77] TechEu (2016). Figure of Bitcoin Mining. <http://tech.eu/wp-content/uploads/2014/03/Bitcoin-proof-of-work.png>. Figure; Accessed 17 Apr. 2016.
- [78] The Economist (2014). Distributed Autonomous Corporation attack. <http://www.economist.com/blogs/babbage/2014/01/computer-corporations>. Online; Accessed 26 Jan. 2016.
- [79] The Economist (2015). How bitcoin mining works. <http://www.economist.com/blogs/economist-explains/2015/01/economist-explains-11>. Online; Accessed 26 Jan. 2016.
- [80] The Film Network (2016). The Film Network's logo. <http://www.thefilmnetwork.co.uk/img/f/logo-gold.png>. Figure; Accessed 08 Jun. 2016.
- [81] Thomas Reuters (2015). Bitcoin bourse: UK's first regulated digital currency exchange in pipeline. <https://www.rt.com/uk/247529-bitcoin-exchange-opening-britain/>. Online; Accessed 17 Apr. 2016.
- [82] Trimble C. (2016). Why online video is the future of content marketing. <http://www.theguardian.com/small-business-network/2014/jan/14/video-content-marketing-media-online>. Online; Accessed 17 Apr. 2016.
- [83] Ujomusic (2016). Ujomusic Home Page. <http://ujomusic.com/>. Online; Accessed 23 Apr. 2016.
- [84] Unknown (2016). Storj Streaming Video off the platform. <http://snart.cc/teststorj/demo.html>. Online; Accessed 21 Apr. 2016.
- [85] visualmojos.com (2016). Icon of user. <http://visualmojos.com/wp-content/themes/visia/images/icons/user-8-icon.svg>. Figure; Accessed 09 Jun. 2016.
- [86] Vorick D. , Champine L. (2016). Sia: Simple Decentralized Storage. <http://sia.tech/assets/globals/sia.pdf>. Online; Accessed 17 Apr. 2016.
- [87] Wiggin LLP (2016). Wiggin's logo. <http://www.wiggin.co.uk/>. Figure; Accessed 08 Jun. 2016.

-
- [88] Wikimedia.org (2016). Figure of the client server model. <https://upload.wikimedia.org/wikipedia/commons/thumb/c/c9/Client-server-model.svg/2000px-Client-server-model.svg.png>. Figure; Accessed 08 Jun. 2016.
- [89] Wilkinson S. (2016). Storj: A Peer-to-Peer Cloud Storage Network. <https://storj.io/storj.pdf>. Online; Accessed 21 Apr. 2016.
- [90] Wood, G. (2015). A Secure Decentralised Generalised Transaction Ledger. <http://gavwood.com/paper.pdf>. Online; Accessed 26 Jan. 2016.
- [91] Zyskind, G. and Nathan, O. and Pentland, A. (2016). Enigma: Decentralized Computation Platform with Guaranteed Privacy. http://enigma.media.mit.edu/enigma_full.pdf. Online; Accessed 23 Apr. 2016.

Appendix A

Innovate UK Feedback

IC Tomorrow: Innovation Contest: Sharing Economy	
Project Title:	PlayRight
Lead Organisation:	Film Network Ltd
Project ID:	72051-428743

How To Read Your Feedback
<p>This document compiles the full comments made by the independent assessors who reviewed your Application. We hope you find it useful.</p> <p>Typically three to five assessors review each application. These assessors are chosen from diverse professional backgrounds and expertise, and each application can be reviewed by both academic and business leaders who inevitably will have different viewpoints. Assessor comments are delivered in their words and presented by question number along with their averaged score. Note as there are multiple assessors involved there are multiple comments made per question and the views of the different independent assessors may not be aligned.</p> <p>We thank you for your application and hope that regardless of the final outcome you find the assessor comments useful in the further development of your project and business.</p>

Scope
<p>Did the assessors consider this application to be in scope? Y</p> <p>.</p> <p>.</p> <p>.</p> <p>.</p> <p>.</p>

Q1. Your Proposed Application
<p>Average score for this section (out of 10): 7.40</p> <p>The video is very clear about the offering and how it will work. The words also back the video up and the concept is compelling and fits the sharing economy.</p> <p>The application explains his proposition clearly, and this proposition is a good fit to the challenge. It has substantial commercial novelty.</p> <p>The applicant describes his vision for film sharing being equitable for all from content producer through to distribution and the viewer, but there is minimal information to say how he intends to achieve that; for example, there is no clear statement explaining why people will want to use this platform and not continue to pirate films.</p>

Fig. A.1 Feedback from Innovate UK Application Page 1.

Although a potentially useful practical implementation of a technology, this represents an implementation of a very long-established model for social media distribution rather than a sharing economy play where there is implicit symmetrical sharing of idle assets. As such, its relevance to the challenge may be marginal.

Undoubtedly an innovative proposal, making use of current trends / media around ~~blockchain~~ technology. Equally the notion of licensed content being shared by fans (whether via a peer to peer model or something centrally licensed/stored) has been explored in the past and faces numerous challenges.

Q2. Proposed Project Plan

Average score for this section (out of 10): 7.80

The plan and breakdown of costs is shown and whilst more info is required and ~~flow of work~~ this stands the project in good stead for potential success. The competition is clear and how this is disruptive to the current way of working and thus difficult not to support.

The project plan is suitably clear on the proposed developments. The cost breakdown is appropriate and is credible.

The project is well segmented into various tasks, some of which seem ambitious in effort terms. Cost has been broken down and a cost is included for the trial. The plan should be able to develop the platform as envisaged.

The breakdown given is reasonable, although does not explicitly address the activities ~~required~~ for a trial with NESTA - ~~any such relationship needs~~ to be inferred.

The costs have been clearly broken down, albeit in amounts that in some cases appear to be rather more on the robust side than expected. Applicants have thought about the components that will be built, together with the number of days it will take to build each item.

Q3. Trial

Average score for this section (out of 10): 6.60

The trials (tests) are explained and the relationship with ~~Nesta~~ clear ~~not be~~ in terms of technology and commercial aspects in terms of payment to see if this works in practice.

The broad concept is appropriate, but the details of the propose trial are not clear. The application would benefit from improving the explanation of the trial, its objectives, and its proposed success criteria.

There is little information on how the trial is ~~expected to be conducted~~. Applicant has identified several measures to gauge success, however target figures that would determine success are not identified. ~~Nesta~~ ~~is expected to be used~~ for wider policy issues. Although not mentioned here, applicant has mentioned that some niche distributors are willing to offer content - how this works for the trial would have been good to describe; identifying a trial partner ought to be considered.

Very little consideration is given to NESTA's remit or goals and to matching this trial to them or of the process for engagement with NESTA.

Fig. A.2 Feedback from Innovate UK Application Page 2.

The applicant has considered which aspects of their service they wish to test. The proposal would be strengthened with more information on what this qualitative and quantitative data might be and how it will be collected. Also details ~~on who~~ will the users be and how they will be acquired.

Q4. Business Model and Marketplace

Average score for this section (out of 10): 6.40

The theory of the pricing model and market is offered in this section - the reality of this in practice has yet to be evidenced - it depends whether those few that have the rights wrapped up fight back over this technology and the cultural change is addressed as well.

Aspect of the business proposition, and its engagement with a substantial marketplace are presented clearly. The application could be further improved by focusing on the real business opportunity in more detail, and it is recommended to move the business spiel further away from complaining about the incumbent studios / distributors, who are successful capitalists whether you agree with their market domination or not.

The market is well described; the current problems that are seen are also described. What the applicant identifies and what is not clearly addressed is persuading viewers to pay for online content when they currently do not. There is no detail on the prices or price points to be charged, the distribution of fees, and the amount that the company would take as a fee/commission.

Although a market model is given, the addressable market and the overall legitimate market growth that would be enabled by any migration of content from unauthorised distribution to legitimate channels is not described. Although it describes a watching fee, no consideration is given here to the revenue split algorithms and therefore to the revenue potential to ~~Playright~~.

An adequate degree of information has been provided around the business model, though some specific numbers on the likely pricing and percentages (even if set by third parties) would be useful. There are some figures on the online video market and the cost of piracy, but more detail on the specific size of the marketplace that this service might likely access would improve the pitch. Especially as the film / TV industry has a chequered history of licensing for online usage.

Q5. Team and Expertise

Average score for this section (out of 10): 8.20

Impressive team that should be able to pull this project off if any combination can.

The applicant's team possesses most of the expertise to deliver the proposition successfully.

The core team has the skills to bring about this development. If skills are lacking, they are not mentioned.

The company will be supported by BAFTA and it is unclear why BAFTA are not cited as the trial partner, as it could be ideal through provision of various content and providing a respected brand for the project.

The team here has relevant skills although identification of skills gaps and how they would be addressed would be ~~useful~~.

Fig. A.3 Feedback from Innovate UK Application Page 3.

The team has been described well, including details of co-founders who have extensive experience within film and TV. From a technical standpoint there is also artificial intelligence and machine learning skills on the team. A little information describing any gaps in the team, or where external suppliers may be needed, would elevate the proposal still further.

Reasons given for not recommending

Fig. A.4 Feedback from Innovate UK Application Page 4.

Appendix B

User Interface Mock-ups

The mock-up shows a web interface for 'ValueChain'. At the top, the logo 'ValueChain' is on the left, and an 'Active Address' with a hexadecimal string '0x359a840a97a0c55ce55092f8238c43789a4264b7' is on the right. The main section is titled 'Create a film'. Below this, there are two steps: '1 Basic Information' (active) and '2 Upload Film'. The 'Basic Information' step contains two input fields: 'Film Name' with the text 'The Revenant' and 'Price per View' with the text '199.0'. A blue 'Next' button is positioned to the right of the inputs. A small downward arrow is centered below the inputs. The footer contains links for 'Get in Touch', 'Terms & Conditions', 'About', a logo with the letter 'L', and social media links for 'Facebook', 'Twitter', and 'LinkedIn'.

ValueChain

Active Address 0x359a840a97a0c55ce55092f8238c43789a4264b7

Create a film

1 Basic Information 2 Upload Film

Film Name

The Revenant

Price per View

199.0

Next

Get in Touch Terms & Conditions About L Facebook Twitter LinkedIn

Fig. B.1 Form for creating a new asset.

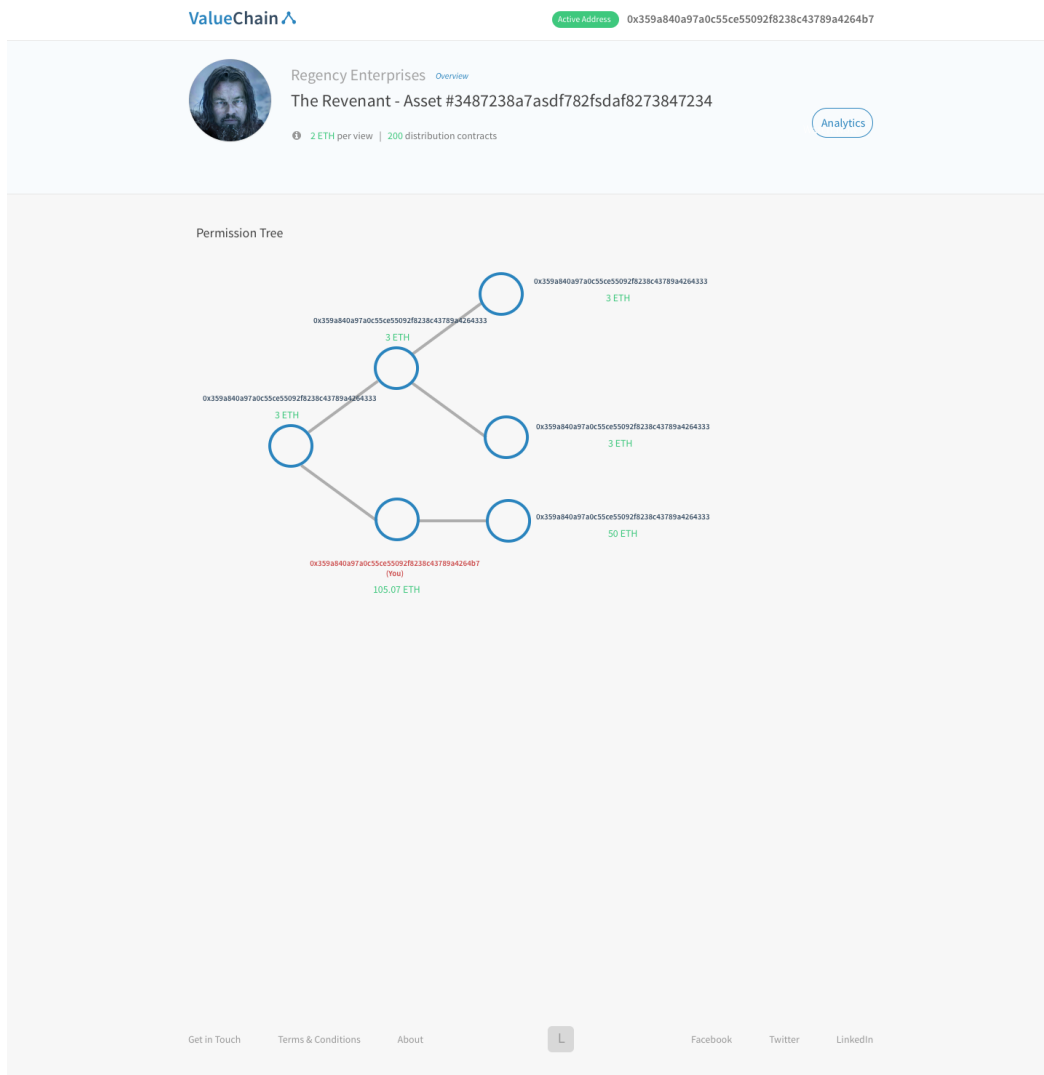


Fig. B.2 The view asset page.

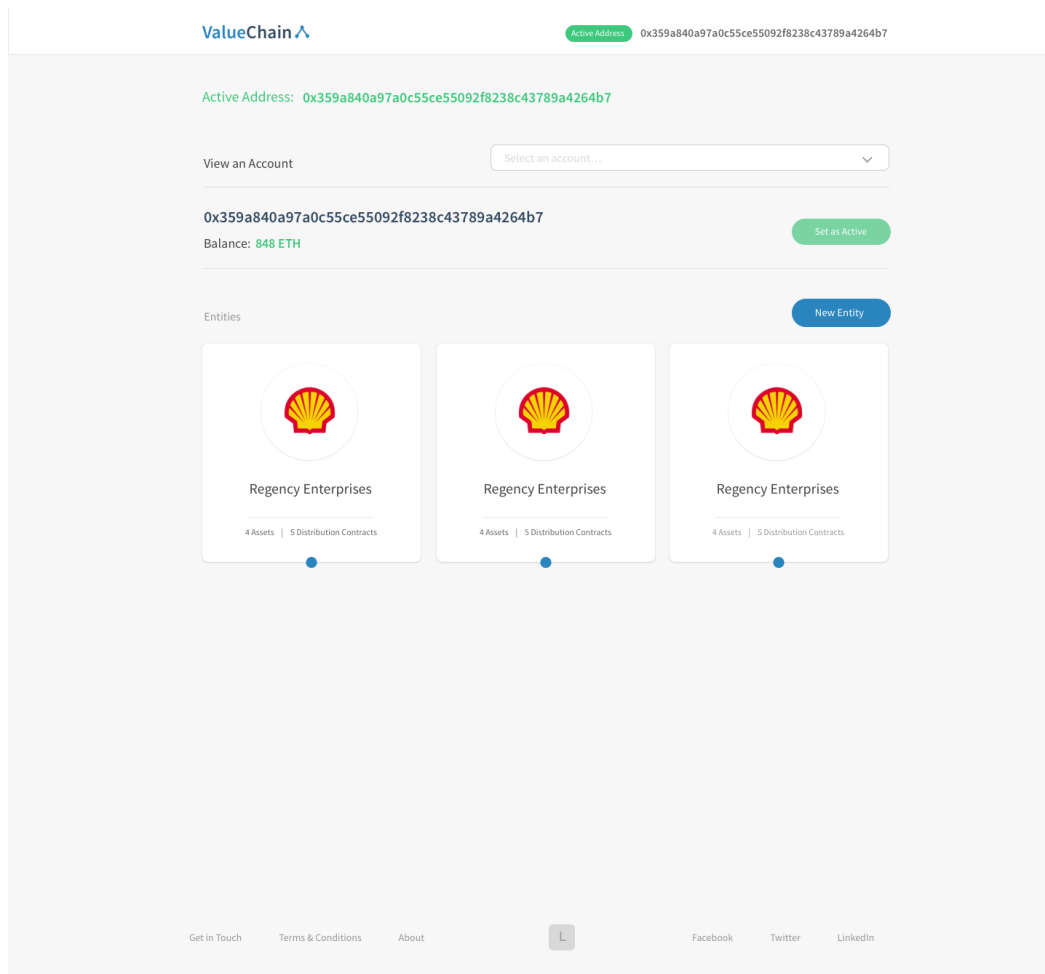


Fig. B.3 The registry/dashboard page.

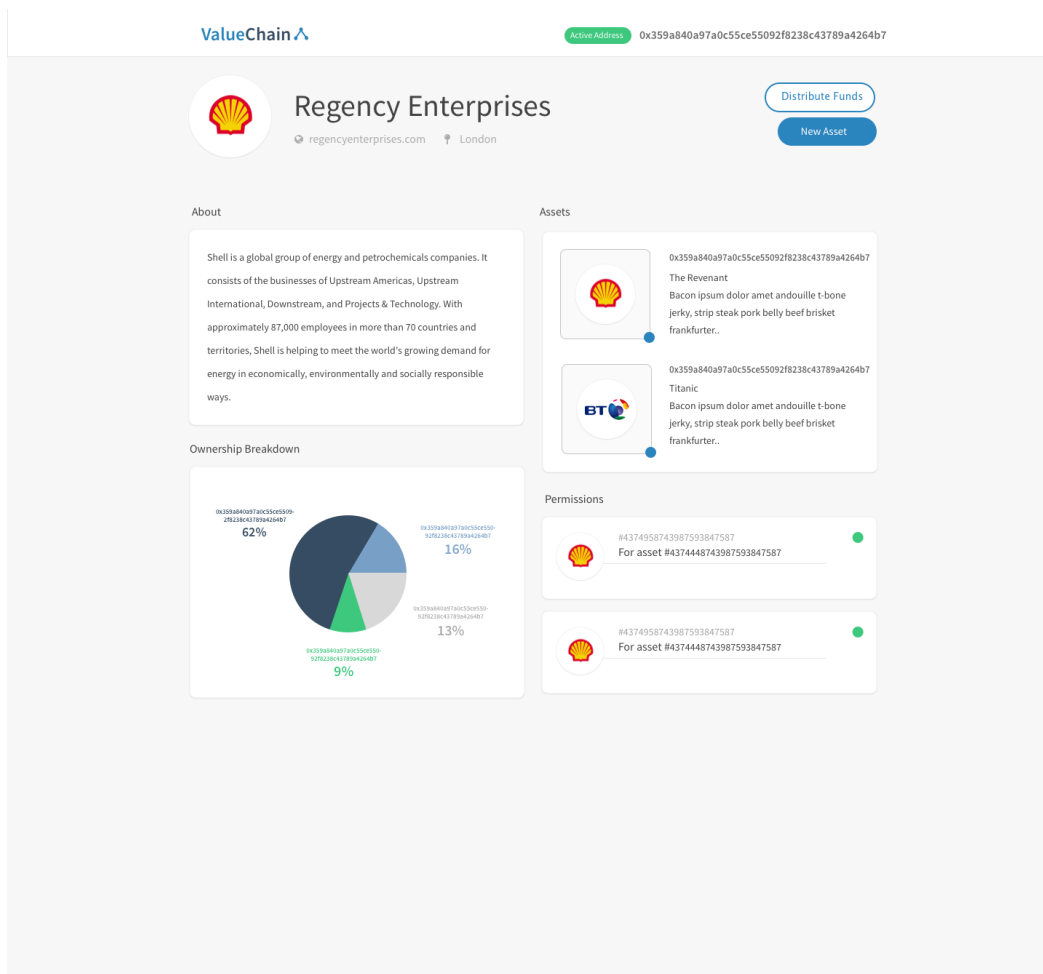


Fig. B.4 The view entity page.



Fig. B.5 The landing page page.

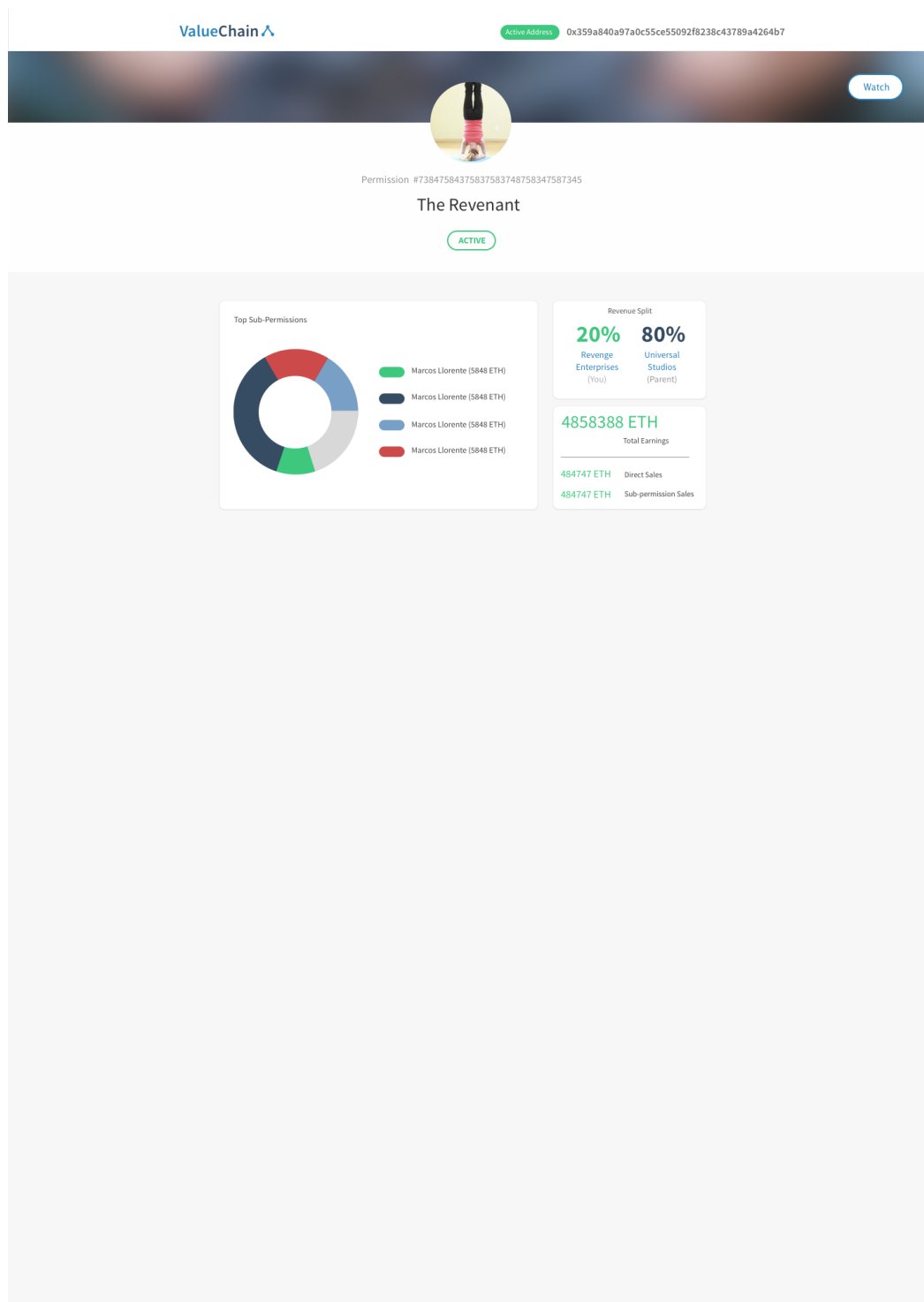


Fig. B.6 The active permission page.

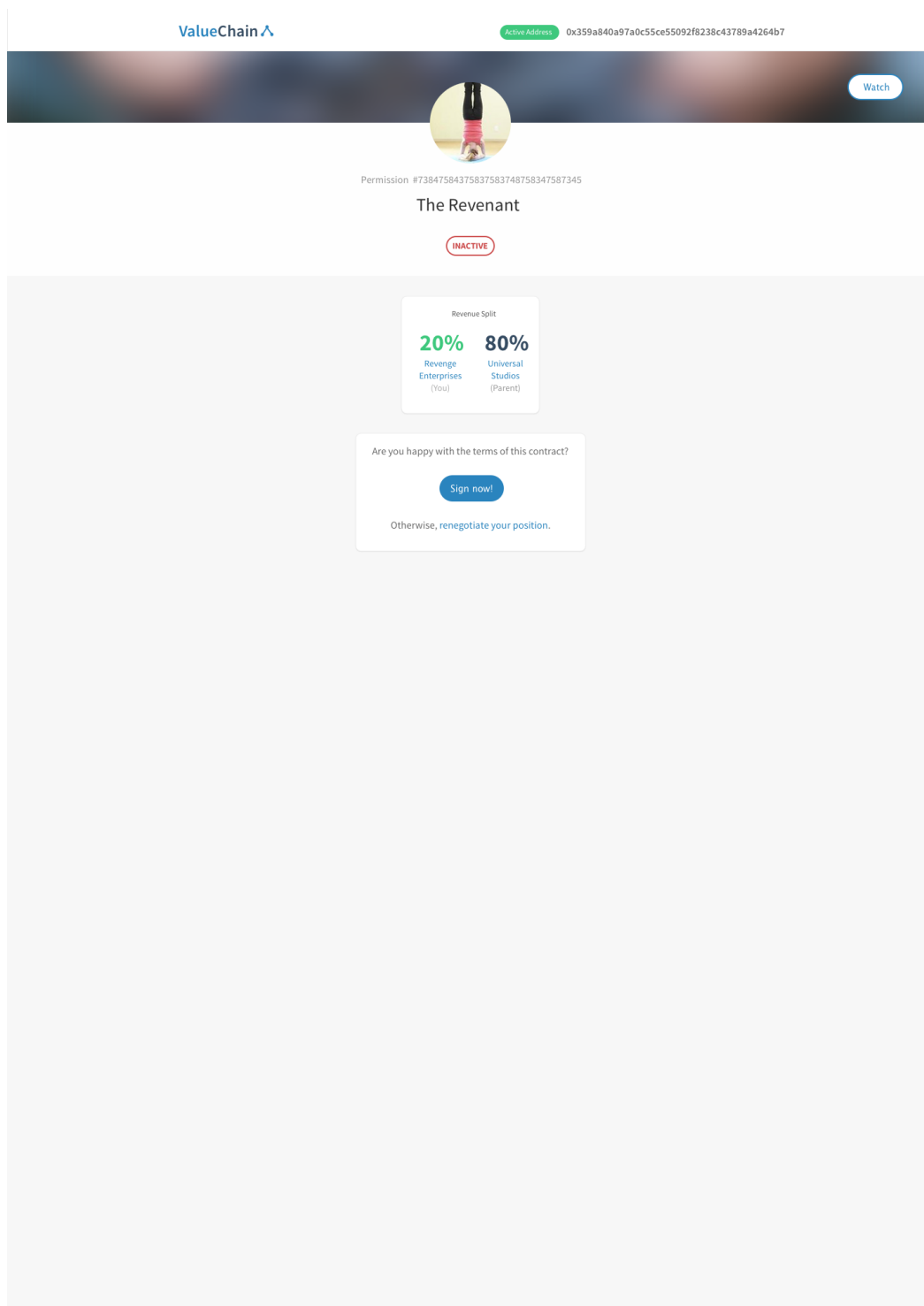



Fig. B.7 The inactive permission page.

ValueChain
Active Address 0x359a840a97a0c55ce55092f8238c43789a4264b7



The Revenant

★★★★☆


7 Views Remaining [Buy a View](#)

While exploring the uncharted wilderness in 1823, frontiersman Hugh Glass (Leonardo DiCaprio) sustains life-threatening injuries from a brutal bear attack. When a member (Tom Hardy) of his hunting team kills his young son (Forrest Goodluck) and leaves him for dead, Glass must utilize his survival skills to find a way back to civilization. Grief-stricken and fueled by vengeance, the legendary fur trapper treks through the snowy terrain to track down the man who betrayed him.

[Sub-Permission](#)


[Stream Now](#)

PERMISSIONING




Build and Negotiate

Build and negotiate a distribution contract with your parent distributor, namely determining the revenue split per view you sell.



Sign

Come to an agreement and sign the contract with your cryptographically secure digital signature. You will then receive a link to a page like this.



Distribute

Customise your page and distribute your link to everyone! If someone buys a film view through your page, you will earn money.

Your parent distributor is [Regency Enterprises](#). Their initial proposal is a 2:3 split in your favour.

Interested?

[Start Negotiating](#)

Fig. B.8 The watch page.