Effective governance of and by the blockchain

Anwaar Ali

University of Cambridge

Computer Laboratory

Wolfson College

Principal supervisor: Professor Jonathan Crowcroft

January 2018

This report is submitted as
the first year's report for Certificate of Postgraduate Study (CPGS)

Declaration

This report is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

This report does not exceed the regulation length of $10,000$ words, excluding tables, figures and bibliography.

Errata

Page	Section	Correction	Corrected on
36	Table 4.1, heading row 8	Changed “Aug 18-Oct 18” to “Nov 18-Jan 19”	29 – 08 – 2018
-	Tiny footnote, every odd page	Changed ”Draft Date: 2002-01-27 12:33:28+00” to “Draft Date: 10-01-2018”	29 – 08 – 2018
36	Table 4.1; row 9, col. 2	Changed “January 31, 2019” to “**January 31, 2019**”	29 – 08 – 2018
20	Eq. 2.1	Changed $\sigma_t \equiv \Gamma(\sigma_{t+1}, T)$ to $\sigma_{t+1} \equiv \Gamma(\sigma_t, T)$	21 – 09 – 2018

Effective governance of and by the blockchain

Anwaar Ali

Executive Summary

This report talks about the problem of of governance in the blockchain ecosystem which is currently a hot issue in the blockchain community. I intend to study this problem by considering two use cases of data ownership and data provenance. After a preliminary literature review it seems that some focus has been given to study the first use case while, to the best of my knowledge, the second use case requires more attention when blockchain is applied to study it. I am also interested in exploring the link between these two use cases to study the overall *data ownership preserving accountable system* which in itself will be a novel contribution of my work. The research proposed in this report may be broadly categorized as a problem related to networked and distributed systems. Specifically, the research that will be carried out will contribute to the topics of system design, consensus mechanisms, algorithmic efficiency, and machine learning methods powered by network measurement data. I will be borrowing a few concepts and well established tools from the fields of security and (maybe also from the) game and optimization theories as well. Ideally, the aim of my PhD is to design an efficient system that addresses the governance issues in the existing blockchain-based systems while considering the use case of data ownership and provenance in an efficient and more systematic manner. My work will also try to facilitate a more systematic approach to study the effective integration of laws and policies with the blockchain technology in terms of smart contracts which is currently also a hot topic.

After a preliminary literature review and following the development of a few blockchain-based projects the aspect of *governance by blockchain* seems quite important and critical. Governance by a blockchain-based distributed system implies an efficient (in terms of time, storage and computation) and trustful (i.e., in an immutable, transparent, auditable and decentralized manner) solution to automate an application in terms of a contract among different non-trusting parties. Such an automation ideally should have the capability to tackle a logical fallacy such as a bug in the code, a malicious activity by an adversary that can often lead to logical and legal ramifications, or a conflict among the members of the contract as well. The absence of such capabilities led to the hack of the decentralized autonomous organization (DAO), a fully automated investment project built on top of Ethereum, resulted in the split of Ethereum community into two blockchain systems. In my observation, immediately after such events the community did not have a predetermined code of conduct to deal with the problem at hand. Currently it seems that the

way these such hacks were counteracted was by deploying a dedicated smart contract. As a result, the whole community was not taken on board during such events thus breaching the original idea of immutability and decentralization of the blockchain technology. If, however, an efficient governance framework were existed to tackle this type of situation then such a split could have been avoided. This motivates me to study the aspect of governance by the blockchain in a more systematic manner.

Similarly, another important aspect is of the *governance of blockchain* system itself. As an example, Bitcoin Cash was born when debate to upgrade the underlying Bitcoin system to make it more scalable arose. This is different from governance by the blockchain as it deals with how to upgrade the underlying protocols and mechanisms of the blockchain technology itself.

My aim is to come up with an overall generic system's solution for the governance problem. The inspiration comes directly from the original premise of trust and decentraliztion of the blockchain technology. This premise implies that this technology can be used for different data record applications. Similarly, my aim is to design a generic and (maybe an) auxiliary solution to the blockchain technology that can facilitate its applications to different use cases while keeping it in compliance with the laws and policies that go with a particular use case. By taking this approach we might be able to preserve the pure decentralized nature of the blockchain idea. In what follows my aim is to review items from the related pool of literature and online articles, discuss different use cases, highlight the events of failures and lessons learned and ultimately and ideally relate these discussions with my own research ideas that has been discussed here.

Acknowledgments

Thank you Jon for being an understanding and accommodating supervisor.

Contents

Chapter 1

Introduction

In this chapter I present the motivation behind studying the topic of blockchain governance. Specifically, I am interested in studying this aspect in terms of considering the use cases of data ownership and data provenance when blockchain paradigm is applied.

1.1 Motivation

The main motivational agents to study governance aspect in blockchain are as follows:

1. Elimination of third parties and the realization of a pure peer-to-peer ecosystem that has the capability to enforce policies and laws in an algorithmic, decentralized, auditable and trustful manner

2. Trust is provided in the form of mathematical and computational evidence

3. A generic underlying technology

4. Absence of a standard framework to tackle the logical and legal ramifications arising during the execution of smart contracts

1.1.1 Lessons learned

The following stories of decentralized autonomous organization (DAO) and other similar hacks, the logical fallacies in smart contracts and the debates on system upgrade that lead to a community split motivates me to study the blockchain governance aspect in a more systemic and detailed manner.

Hack story: Decentralized autonomous organization (DAO)

The decentralized autonomous organization (DAO) was a smart contract built on top of Ethereum (discussed in Section 2.3.1). The DAO automated a venture capital fund for investors to vote on the proposals for star ups. The DAO also provided investors with an exit door implemented using a *split function* to opt out of an investment if they thought that the project proposal was not good enough. Overall The DAO project managed to attract $\$150M$ worth of Ether (underlying cryptocurrency in Ethereum) in investment.

The DAO was later hacked by exploiting the recursive call of the split function[1]. The fallacy was whenever the split function was called an investor was able to withdraw the investment first and the overall balance of The DAO was updated later. This way the hackers called the split function multiple times to drain about $\$70M$ worth of Ether before The DAO could update its balance. This was possible because the developers of The DAO did not consider the possibility of the recursive call and because of The DAO updated its balance after giving the funds back to the investors who wanted to opt out.

The solution that was proposed and implemented to reverse the effects of this hack resulted in the split of Ethereum community. A *hard fork* solution was implemented by designing and executing a smart contract whose function was just to withdraw the funds from The DAO and give them back to the investors. Those in the opposition of this solution later started Ethereum Classic[2]. Their argument was that such second chances breached the very notion of decentralization and immutability that blockchain promised. According to their philosophy even the bad memories should be kept and system should be allowed to heal on its own. Further can be read in Ethereum Classic's detailed manifesto[3]. It is important to note here that the fallacy was not in the Ethereum platform itself but rather the way DAO's smart contract was programmed. The lesson to be learned here is to adopt good and well tested strategies to program a smart contract[4]. In my research I will give this issue full consideration.

Hack story: Stealing of $31M Ether

This hack was another instance of a bug in software. In the cryptocurrency world a *wallet* software provides a window to the underlying blockchain and to manage one's digital assets. A wallet is usually accessed using one's private key (see Section 2.1.2). A specific type of wallets is called *multi signature wallets* where more than one public key is usually required to unlock it. Such a wallet was hacked by exploiting the vulnerability where one bug allowed anyone to reinitialize the wallet software and all the keys associated with it

[1] `https://www.cryptocompare.com/coins/guides/the-dao-the-hack-the-soft-fork-and-the-hard-fork/`
[2] `https://ethereumclassic.github.io/`
[3] `https://ethereumclassic.github.io/blog/2016-07-11-manifesto/`
[4] `https://consensys.github.io/smart-contract-best-practices/`

[5]. \$31M worth of Ether were stolen. The solution that was deployed was in the form of a smart contract that hacked all the remaining wallets before the hackers could and deposited the money in safe address which was latter returned to the related owners of the funds. But one important thing to note here is that this hack, unlike the last one, was irreversible i.e., the \$31M were gone for good. Such a fiasco along with the fact the community had to write a dedicated smart contract on the spot motivates me to study the blockchain governance aspect in a more systematic and detailed manner.

Hack story: IOTA

Here the vulnerability was in differential cryptanalysis that was discovered by Digital Currency Initiative (DCI) at MIT[6]. Reading about this problem there seems to be a severe lack in the standardization of the cryptocurrecy deployments. Specially in security aspects of things. Take away message is that it is a good practice not use custom cryptographic methods rather use the ones that have been in practice and vetted by the community over the years. Given the rapid growth of cryptocurrencies it is a daunting task for the community to vet all the code of such a system. There might be a silver lining if a framework is developed powered by techniques such as formal verification of code like the one Tezos uses in its system (see Section 2.3.4).

Bitcoin split

The split in Bitcoin that resulted in new currency system called Bitcoin Cash[7] was the result of a heated debate to address the scalability issue [8]. In my opinion such a split can be avoided if decisions are made in a decentralized manner as well by automating an efficient voting process on top of blockchian itself. This governance aspect is envisioned by Tezos by introducing the concept of formal verification of code (as mentioned above). In my research I want to extend this notion to build an more generic governance framework for blockchain.

1.2 Personal data ownership and management

In this data rich age one signs up for many software applications and makes an agreement with third party service providers and allows the use of one's data for personalized services. These services can take the form of personalized recommendations e.g., on websites like Amazon and IMDb. There is also a concern of third parties using or selling a user's data

[5]`https://medium.freecodecamp.org/a-hacker-stole-31m-of-ether-how-it-happened-and-what-it-means`
[6]`https://medium.com/@neha/cryptographic-vulnerabilities-in-iota-9a6a9ddc4367`
[7]`https://www.bitcoincash.org/`
[8]`https://www.coindesk.com/bitcoin-cash-hard-forks-blockchain-bid-ease-mining-difficulties/`

for selfish, wrongful or for spying purposes. This raises the issue of user privacy in such data powered soft-systems.

A recent digital economy bill in the UK was the topic of debate [2]. The debate was over the data gathering from multiple sources to track a select group of people (e.g., immigrants using the UK's National Health Service (NHS)). The question is whether the owners of this data know of this snooping? Is it even legal for the UK (or any other country in a similar way) to do so?

Another case that can be considered for the motivation is a simple sign up process on many web and mobile applications. If a user, along the way, does not feel comfortable with the type of personal information sharing she agreed on while sign up the only choice that remains with her is to re-sign up. There should be a system that allows users to control and own their data all the time yet at the same time giving third parties a set of permissions to use their data. This set should be modifiable on the go with the extreme case of this set being empty, i.e., a user can revoke a service/application's right to access her data at anytime she wants. In [25] blockchain is presented as an access control manager. A user's data sits in an off-blockchain storage (such as distributed hash table (DHT) database) and permissions related to access of this data by any sort of third parties is set by a user by making a transaction to the blockchain. This way the data access information for a piece of user data is mined in a distributed and trusted manner in a block of a blockchain.

Such a an access control system provides *second chances* for the owners of the data to either modify or completely revoke the permissions related to their data. These permissions can be modified/revoked in a per-service manner or in a collective way (same action that affects permission level of all the services at the same time).

One extension to the work [25] as its authors mentioned can be to only return the computations on one's data and record it on top of blockchain. This means third parties are not given access to the raw data of a user (which they can store locally for later use) rather only the result of using this data is published and recorded on blockchain. This way if a third party tries to behave maliciously the record of this computation may be presented in a court of law to prove this behaviour. Another advantage of this could be to demystify the workings of an algorithm on users data. When a computation on one's data is returned one can see what and how one's data was used. Since this activity is recorded on blockchain there can be a system to incentivise users who grant permissions for third parties to use their data.

1.3 Data provenance and system audit

Data provenance implies keeping a track of the metadata that records the history of creation of a piece of data and different operations that are performed on it during its life cycle [7]. Efficient data provenance systems help realize auditable computation systems [21, 22, 19]. Tracking the provenance of a piece of data can help detect different anomalies, break ins and data access violations thus improving users' privacy and integrity of data. Data provenance is particularly important in current age of cloud computing where most of users' data sit in a storage facilities of third parties. Even if the third party cloud storage providers implement a data provenance system the users still have to trust the third parties in their claim that such a system is trust worthy. If, however, we apply the blockchain paradigm to implement a data provenance system then a distributed and trusted data provenance system might be possible.

ProvChain in [16] implements a data provenance system for cloud storage using blockchain. ProvChain considers a file as a unit of data and tracks the operations of creation, sharing, modification, and deletion of a file object using blockchain. An operation on a file object is recorded as a node of Merkle node (see Section 2.1.2 for an explanation of Merkle Tree in blockchain). A Merkle root for a set of operations on a file object is then stored in a block of underlying blockchain as a transaction. A separate database is also maintained to record all of these operations on a file object. An auditor is then provided with a blockchain receipt containing the information of block number and a transaction id that tracks a certain set of operations that the auditor is interested in. Since such a block is mined using a distributed consensus protocol a user does not have to trust a third party for the integrity of provenance data. If there is a mismatch between the record extracted using a blockchain receipt and the corresponding provenance data stored in a separate database then the auditor can raise an alarm for a possible break in. This work talks about one cloud service provider with one application. I intend to study its extension to multiple cloud providers with different applications in combination with the data ownership use case as discussed in the last section to study the overall ownership/privacy preserving auditable system. Besides storage, another interesting work is presented in [9] that studies the verifiable cloud computing using smart contracts. This work makes use of game theoretic techniques to study the aspect of collusion between two cloud providers for a same computation task. When both the providers return result a cross check can establish the existence or absence of collusion between them. I am interested in studying how efficient, in terms of overhead, it is to implement these systems alongside simple cloud storage and computing.

1.4 Philosophy of second chances

Learning from DAO and another hack there seems to be further two types of second chances: completely reversible like the one given to DAO investors and the other one not reversible like the money stealing hack.

The discussion so far mostly seem to imply that one wants to keep all the third parties at an arm's length. In my personal opinion, however, I do not see all the third parties as pure evils. At the end of the day we do need recommendation systems and personalised services like healthcare. It should be more about reconciling users and third parties in a more secure, reliable, auditable and trustworthy manner. There should be second chances for the third parties as well. This could be based on reputation gauging mechanism like discussed in Section V-B of [25]. Since we are thinking along the lines of automating a trustless, immutable and auditable system using blockchain the idea of reconciling users and third parties can also be realized at least up to some extent.

Chapter 2

Background

I will largely base my discussion in this chapter on the seminal whitepaper of Bitcoin [18] which was the original incarnation of these concepts. Along the way I will briefly discuss how these concepts can be generalized and extended to other systems and use cases as well.

2.1 Blockchain

The blockchain technology establishes trust in a distributed and peer-to-peer (P2P) networks without the interference of a third party. Records of transactions among the peers of a P2P network are stored in a chain of blocks called the blockchain. Each peer of such a P2P network maintains a copy of this blockchain and hence it is sometimes referred to as the *distributed ledger technology (DLT)* as well. Each copy of a DLT gets updated at the same time with no provision of retroactive mutations in records.

The blockchain technology realizes the notion of *trustlessness* among the peers of a P2P network. Since the original incarnation of blockchain was financial transactions, the trustless property in this use case implies that two parties can make a financial transaction without either trusting each other or even without the knowledge of other party's identity. This encourages anonymization where each party can simply use a pseudonym instead of her real identity. Immutability of transactional records, which are recorded in different blocks in a blockchain, is another lucrative feature.

2.1.1 Hashing chains the blocks together

A hash is a one way mathematical cryptographic function. Given an input of any size and form a hash function always generates a random fixed length output string of characters. Each output of a hash function is unique to the input given. If the input is changed

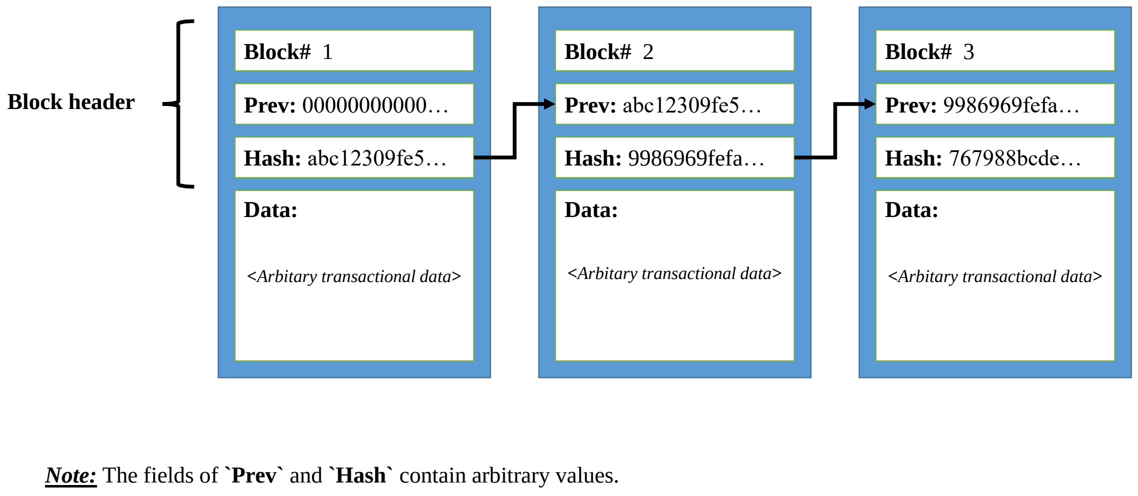

Figure 2.1: Structure of blockchain: hashing chains the blocks together

the output almost always changes randomly and completely. This way output of a hash function can be considered as the integrity test of some data. Each block in a blockchain contains a data field that stores transactional data among peers of a network along with one field which contains the hash of all the contents of this block and another field that stores the hash of the previous block. Inclusion of the hash of a block in the following block chains the blocks together. By doing this a change in a block is reflected in the portion of the blockchain that comes after it. This hashing scheme along with distribution of this chain among the peers of the network gives blockchain its immutability aspect. These features of blockchain make a third party redundant which would usually be required to facilitate a transaction between two parties. Fig. 2.1 shows the structure of a blockchain. Fig. 2.1 shows that the further back an attacker tries to tamper with a block the longest portion of the chain she will disturb. Such an attacker then has to rehash all the blocks following the one where she does the tampering. Now since there are other copies of this chain distributed throughout the network, such an attacker has to do the same tampering with at least 51% of the copies to make this change acceptable in the network. This is how the immutability aspect of blockchain is realized.

2.1.2 Transaction chain

It should be noted here that the transaction chain is different from the actual blockchain. A block in a blockchain can contain multiple transaction chains which are also alternatively called coins. Fig. 2.2 shows a coin. While moving from one owner to the next the length of the coin (or a transaction chain) always increases. This coin might take any form and

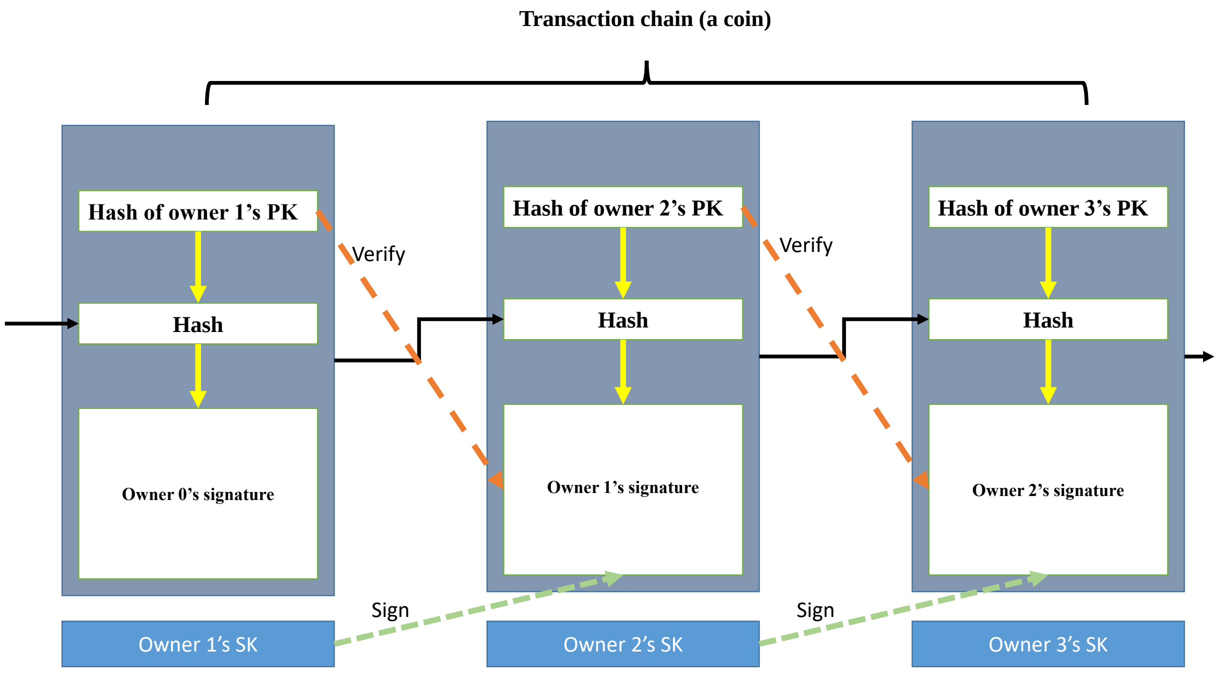

Figure 2.2: Transaction chain or a coin. Reference: [18]

meaning. We can define any sort of asset as we please. Generally such a coin is referred to as a token. As an example, a blockchain-based platform called Ethereum gives us the provision to define a token of our own[1]. As shown in Fig. 2.2, public key infrastructure (PKI) is used to make a transaction in blockchain ecosystem. A party digitally *signs* a transaction with its private key (SK) and sends it to the address of the payee which is basically the hash of its public key (PK). In order to conserve storage, a set of transactions are hashed as the nodes of a Merkle Tree [17] structure and only the root of a Merkle Tree is then included in a block [18].

2.2 Consensus protocols

There are two main categories of consensus protocols being deployed in the blockchain-based systems [3] namely:

1. *Lottery-based protocols:* Such as proof of work (PoW) [18] and proof of elapsed time (PoET) [2].

2. *Voting-based protocols:* Such as Byzantine fault tolerance-based methods and Paxos [14, 15].

[1]https://www.ethereum.org/token

[2]https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html#proof-of-elapsed-time-poet

First, I will explain, in detail the PoW which is the most poupular choice by the blockchain systems so far and then briefly discuss a few other consensus protocols that are being considered by the community as well.

2.2.1 Proof-of-work (PoW)

PoW is the lottery-based consensus protocol introduced in the original Bitcoin whitepaper [18]. PoW is presented as the solution to the *double spending problem.* Double spending implies a non-ethical behaviour by an owner of a coin (or, in general, a token) in the transaction chain. Such an owner tries to make more than one payment to different payees with the same amount of assets that it owns. PoW is a mechanism in which a payee is provided with a *proof* that enables it to establish the absence of double spending. The proof is actually shown in the form of the hash of the block. The hash output of a block must be in a particular form. In Bitcoin's case, this output should start with a predefined number of zeros. In order to obtain this format of the hash of a block, some amount of work is required. Here the *work* implies CPU consumption in terms of time and electricity which is also referred to as *mining.* The nodes of the P2P network that do this work are called the *miners.* Specifically, mining is when a CPU guesses a number called a *nonce* that would render a block's (with all the details in it—like transactional data, next owners address and current owners digital signature) hash to start with a predefined number of zeros (see Fig. 2.3). This predefined number of zeros is actually called the *difficulty* of the mathematical puzzle of guessing a nonce. This difficulty increases exponentially with increasing number of zeros. In order to accommodate the increasing processing speeds of computers the level of difficulty can be increased accordingly along the way by simply putting a constraint of larger number of zeros in the hash output[3]. A miner node usually collects a few transactions in the P2P network and starts to guess a nonce that will ultimately achieve the predefined number of zeros with the contents of the newly collected transactions. As soon as a miner finds a nonce it broadcasts this solution in the form of a block. After this step, consensus takes place. Nodes in the network validate the transactions contained in this block (in terms of checking the double spending instances) and an acceptance of this block is manifested if the network starts to create a new block by using the hash of the newly created block as a part of the new block (this is referred to as the *Previous hash* field in blocks of a blockchain). The miner node is rewarded in terms of newly generated bitcoins.

2.2.2 Proof of stake (PoS)

Proof of stake (PoS) is being considered as the next step from PoW. In PoW the consumption of electrical and computational resources is an issue. There is also a chance

[3]https://en.bitcoin.it/wiki/Difficulty

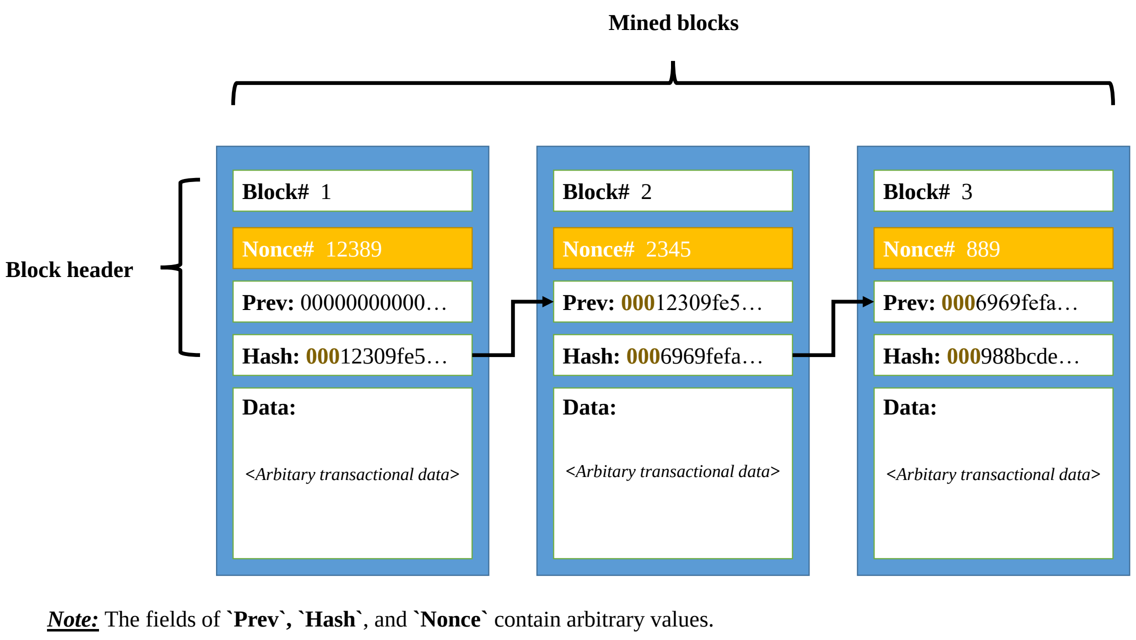

Figure 2.3: Chain of mined blocks. Notice that the 'Hash' of each block now starts with three zeros.

of centralization in terms of one party or a group of parties owning more computational resources than others. In PoS-based consensus the actual digital currency (or tokens) of the underlying system will be at *stake* instead of electrical and computational resources. In PoS-based system a set of nodes called *validators* propose and create new blocks in turns[4]. An elected validator node puts forward a certain amount of the underlying cryptocurrency as a deposit which is then recorded and locked in the blockchain. This deposit determines the weight of the votes that validators cast to select a new block. Once a block is created and proposed other validator nodes then reach a consensus on this block. This consensus can be in a traditional manner like it was done in the PoW-based blockchain systems or based on Byzantine fault tolerance-based methods as discussed later. Among other consensus protocols that are being deployed in blockchain-based systems are proof of elapsed time (PoET)[5] in which a leader is elected with the shortest weight time among the randomly assigned weight times, and Byzantine fault tolerance-based mechanisms [14].

[4]https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ
[5]https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html#proof-of-elapsed-time-poet

2.3 Evolution of blockchain ecosystems

2.3.1 Case study: Ethereum

Ethereum[6] [24] provides a decentralized platform that can be used to deploy decentralized value transfer applications or *Dapps.* These Dapps are famously called *smart contracts* that run exactly as specified in their code without any third party interference or censorship. This is possible as these smart contracts run on top of a blockchain.

Smart contracts

In Ethereum, smart contracts can be referred to as the algorithmic enforcement of agreements among, often non-trusting, parties [24]. The whole Ethereum platform enables one to implement an application use case in terms of a smart contract among different parties that can collectively be considered as a state machine. This machine changes its state depending upon transfers of value exchanges among parties. Due to the underlying blockchain technology, the record of these state changes are recorded in a decentralized, immutable and transparent way. This enables two parties, that would not have trusted each other, to make a transaction with each other without the involvement of any third party. A state can consist of different sorts of records such as account balances, operations on a piece of data (as discussed in Section 1.3), agreement specifics among the parties (such as access control instructions as discussed in Section 1.2) etc. A transaction among parties triggers a state change for example from state σ_t to σ_{t+1} according to a state transition function Γ according to the following Eq. 2.1 (reference [24]).

$$\sigma_{t+1} \equiv \Gamma(\sigma_t, T) \tag{2.1}$$

The above Eq. 2.1 enables the nodes in the Ethereum ecosystem to carry out arbitrary computations which can be specified by the function Γ and then store arbitrary states between transactions in a decentralized fashion which is denoted by σ. The transactions among parties that trigger a state change are then recorded into blocks in a manner similar to the one described in the last Section 2.1. Currently, Ethereum uses PoW-based consensus as well, however, its community is considering a shift to PoS[7]. The Ethereum protocol that handles the state transition and computation is also referred to as *Ethereum virtual machine (EVM).*

Similar to as described for Bitcoin, the entities in Ethereum also have addresses through which they can be accessed. However, there are two main categories of entities in

[6] `https://ethereum.org/`

[7] `https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ`

Ethereum's ecosystem[8]. One of the entities is called an *externally own account (EOA)* and the other entity is the smart contract itself which is also assigned an address that can be accessed either by an EOA or by another smart contract. Anyone, using an EOA, can transfer value with another EOA by specifying its address. In this case the overall state of the system remains the same. It is, however, also possible that a transaction by an EOA is addressed to a smart contract. This way a certain function in the smart contract's code can be triggered that leads to a state change in the overall EVM. It can also happen that a transaction triggers an action in one smart contract which in turn invokes another smart contract, possibly executing another EVM.

One important thing to note here is that blockchains are only used in this perspective to provide correctness and availability of transactions. Privacy, the particular terms and conditions of a contract and contractual security still need to be programmed on top of a blockchain. Hawk [11] is a platform that can be used to write contracts in a high level language in an intuitive way. The Hawk compiler then implements this contract in a cryptographical manner on top of a blockchain. Two types of security guarantees are provided in Hawk i) on-chain privacy and ii) contractual security.

i) On-chain privacy: Hawk cryptographically hides the transactional data of the parties involved in a contract onto a blockchain. This is done so that only the parties involved in the contract are aware of the actual data of their transaction and yet at the same time the underlying blockchain is being used to ensure the correctness of the transactions. This way if a party wants to abort it may have to pay for it.

ii) Contractual security: This guarantee implements fairness and protects the interacting parties from each other (in contrast to the outside world like discussed in the previous paragraph).

2.3.2 Case study: Hyperledger

Hyperledger[9] is an infrastructure project by the Linux Foundation[10]. It consists of multiple frameworks and tools. The aim of this project is to provide cross-industry blockchain solutions with the model of openness for development and collaboration of its architecture. The distinguishing features of Hyperledger project consist permissioned architecture, modularity, focus on consensus, focus on businesses with the focus on vendor-neutral-permissioned-blockchain networks and interoperability of blockchains and their different components. The plug-and-play aspect of this project is very lucrative which enables one to try different versions of a component of a blocckhain such as different consensus protocols. This provides the opportunity to conduct a comparison study among different consensus protocols for the same underlying use case.

[8]`https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial`

[9]`https://www.hyperledger.org/`

[10]`https://www.linuxfoundation.org/`

Permissioned blockchains

Initial blockchain-based systems such as Bitcoin and Ethereum are sometimes referred to as the permissionless system. This implies that anyone can join the underlying P2P network of blockchain. Anyone is free to make transactions with whomever one desires. However, permissioned approach implies that a trusted third party decides on the membership of nodes in terms of who can join the network and make transactions with other trusted members. Here the blockchain is present to reinforce the trust in the trusted third parties that manages the membership mechanism. The blockchain itself can be available publicly in order to bolster the transparency of operations and trust [5].

Hyperledger framework

The Hyperledger framework consists of following components:

Sawtooth: Provides a modular platform to build, deploy and run distributed ledgers. It uses the consensus protocol Proof of Elapsed Time (PoET). As discussed earlier, PoET tries to achieve minimal resource consumption by taking into consideration the large distributed populations of validators.

Iroha: It can be used for an easy and simple incorporation of distributed ledger technology into the business infrastructures.

Fabric: With the lucrative modularity feature of *plug-and-play* for different blockchain components such as consensus and membership services this is a framework for application development. It makes use of container technology (like Docker[11]) to host smart contract code which is also called *chaincode*.

Indy: This framework is a collection of tools, libraries and and reusable components to help provide identities based on blockchains and distributed ledgers and make them interoperable with each other for others (from different administrative domains and fields) to use them effectively.

Burrow: This framework provides a node to run *permissioned* Ethereum smart contracts. Specifically this node consists of three components: a consensus engine, permissioned Ethereum virtual machine and a gateway to rpc.

Hyperledger tools

Hyperledger providers different tools with specific functions and capabilities:

[11]`https://www.docker.com/`

Composer: This tool provides a collaboration platform to build business blockchain networks, smart contracts and in turn their deployment on the distributed ledger platforms.

Explorer: This tool provides means to explore different types of data of the distributed ledger e.g., blocks, transaction data, network data and chain codes.

Cello: A tool provided to realize the on-demand and *as-a-service* aspects in the blockchain ecosystem. This helps ease the processes of creation, management and termination of blockchains. **Q:** If it indeed eases these aspects then can multiple blockchians easily be deployed? This idea specifically came to me after reading through Chris Reed's *Beyond Bitcoin* paper. Moreover, if this is easy and the overall infrastructure provides one to use different blockchain components like consensus in a *plug-and-play* manner then using multiple blockchains and then interoperating them might be a relatively easier task. This specific aspect can further be studied in light of insights gains from going through the details of Hyperledger and Chris Reed's paper. Combining the concept of *merging two blockchains* with my concept of OSC could be a lead to build on my premise of developing a legal-ramifications' framework.

2.3.3 Case study: Zcash

Like Bitcoin, Zacash is a decentralized, permissionless, and opensource cryptocurrency. The distinguishing feature of Zcash[12] is the implementation of privacy. The aspect of privacy here implies that the identities of a sender, the recipient and the transactional amount between these two parties remain private. The privacy implementing mechanism, however, is capable of establishing the validity of a transaction.

The privacy implementing mechanism in Zcash is based on the cryptographic technique called the *zero-knowledge proofs*. In this technique a prover is able to prove a claim to a verifier without revealing the actual contents of the claim itself. A simple example of this can be that a prover can prove to a verifier that she knows a certain number with an associated hash value without revealing what the actual number is.

The Zcash makes use of zk-SNARKs which stands for *zero knowledge succinct non-interactive argument of knowledge.* This enables a prover of certain information to prove a claim about this information to a verifier in a light-weight (in terms of memory which is in the order of a few hundred bytes) and non interactive manner. The *non-interactive* property implies that a prover does not need to have a message exchange in a back-and-forth manner with the verifier and one interaction would be enough to prove a certain claim. This is achieved by setting up a random reference string between prover and the verifier during the initialization process. This randomness process must be kept secret

[12]`https://z.cash/`

and secure otherwise an adversary with an access to it can prove false information to a verifier. This implies generation of the counterfeit coins. More in depth information can be looked at by consulting the white paper [20]. This work motivates me to use a similar zero-knowledge cryptographic technique to implement the privacy layer on top of an application while is being automated using smart contracts on top of a blockchain.

2.3.4 Case study: Tezos

The main purpose of this case study is to the highlight the two distinguishing features of it specially the one that relates the most to my concept of *overarching smart contract (OSC).* Right now I have a hunch that Tezos[13] too might need a formal framework to implement the governance aspect that it proposes.

Their concept of *commonwealth* is quite interesting. This might have been used because the reward system is also in terms of the internally generated currency (or tokens) and the overall decision is also made by the relevant (internal) community as well on the blockchain. But it still does not answer the question of what if something goes wrong while a smart contract is being deployed and is currently in execution? However, this system seems quite important to take motivation from. Still it seems like it provides a way to upgrade a system, democratically—which seems to be the lucrative aspect, but only after something goes wrong and not during the faulty execution of a system.

Specifically, there are two aspects that make Tezos distinct from existing blockchain-based systems, in particular from Ethereum and those are:

1. Its self-amendment aspect. This very aspect led to a split in Bitcoin and we will (perhaps) soon see an upgrade from PoW to PoS in Ethereum as well (although, I am speculating there will be another round of heated debate before and if it happens). These have been hardforks in these systems after usually a heated and prolonged debates which sometimes have resulted in the split of the overall community. Now with Tezos if an update is required (or a malicious activity takes place in the system) then it can be achieved (or counteracted I presume) using blockchain itself so that things do not run amok (that has been the case so far, as for example what happened when DAO on Ehtereum got hacked that led to the split in Ethereum and Ethereum Classic) and an upgrade is performed by keeping the full community in the loop. So I see it as kind of a *framework* that allows you to make decisions about the underlying system itself in a systematic and democratic manner. There is also the lucrative feature of such activities being rewarded which keeps the interest of the community in the system and encourages them to take the initiative to either improve the system or to report something faulty like a bug or a vulnerability.

[13]`https://www.tezos.com/`

2. The second part is concerned with formal verification of the software code [10] that runs on top of a blockchain. Now, as an analogy it is quite similar to the notion of trust in the original blockchain concept. This trust isn't the human good will but a mathematical proof obtained, for example, by using PoW consensus protocol (i.e., it is precise and verifiable—you just have to run the hash function). Similarly formal verification of code makes the whole system of Tezos quite secure (as compared to I think Ethereum and Bitcoin's designs) as now you do not need to have hard and fast checks on the code to check its correctness (i.e., the code will do what it is supposed/claimed to do and will not lead into some sort of unexpected outcome—I presume; so the concepts such as *gas* in Ethereum might get redundant). This is another kind of trust where you do not necessarily have to believe a software developer's claim of her code to be correct you can just verify it mathematically yourself. In this way it is, for sure, distinct from Ethereum (and more secure in turn as well)

2.4 Use cases

2.4.1 Financial

It is important to discuss this category of use cases as it was the first area the blockchain was proposed for. Besides the use case of cryptocurrencies (which there are numerous these days) there are other use cases in this realm as well. One example is of insurance as one of the blockchain services provided by KPMG. Smart contracts can be used to settle claims and make payments in an economical fashion as an adjudicator can be eliminated and the whole process can be automated in a trusted manner [1].

2.4.2 Technological

Following two projects are quite interested:

Blockstack[14] [4]: Discourages centralization and eliminates third parties such as centralized DNS servers. Through this, it enhances user data ownership, privacy, and freedom. Namecoin is another similar project[15].

Filecoin[16] [13]: Implements decentralized storage network. It reward anyone who share her storage in a decentrazlied manner.

[15]`https://namecoin.org/`

2.4.3 Governmental

The report [23] shows the interest of the UK's Government in implementing blockchain to make various of its projects fair and transparent such as voting and identity management. Identity management in particular is quite important for many of the planet's less developed regions. With blockchain a central and trusted identity management system can be established. Inspiration can be taken from Estonia's case which deploys blockchain to makes many of her processes fair and transparent[17].

2.4.4 Medical

Health record management where a user controls and owns her data at all times. Major issues to consider in this use case are:

1. Data fragmentation
2. Data privacy and ownership—also sometimes referred to as patient agency
3. Interoperability (although it could be the same as data fragmentation)

These issues can largely be resolved using a blockchain-based health record management system as envisioned by MIT's MedRec[18].

2.4.5 General provenance tracking

Supply chain and energy sector.

[17] `https://e-estonia.com/`
[18] `https://www.media.mit.edu/research/groups/1454/medrec`

Chapter 3

Proposed research problem and philosophy

3.1 Research problem space for doctoral investigation

3.2 Governance framework

I envisage a blockchain governance framework as shown in Fig. 3.1. It can be seen that there are four main interacting layers namely:

1. *Formal verification of code:* As envisioned by Tezos [10], this layer can be implemented to attach certain guarantees to the smart contract code developed after the deliberation on human-policy making level. The main aim of this layer is to decouple, as much as possible, the human logic and decision making to its implementation in the form of smart contract code. The purpose is to make the code development as trust worthy as possible while mostly holding the actual human thinking accountable. This might pave the way to using the traditional legal proceedings in case something goes wrong in the system.

2. *The smart contract layer:* This layer further has two sublayers for overarching smart contract (OSC) and underlying smart contracts (USCs). The overall intelligence to automate a blockchain ecosystem is implemented by this layer. The OSC sublayer may interact with the underlying USCs by issuing different *policy enforcement instructions (PEIs)* while the USCs automate a use case at hand (such as data ownership and data provenance). We may also be able to push updates to improve/modify the logic behind the governance of a use case or of the blockchain protocol in general to USCs through OSC.

3. The next layers reflect a certain use case at hand. Depending upon the nature of the use case we might need an *auxiliary item* such as an off-chain storage solution as proposed in [25] and [16] and discussed in Sections 1.2 and 1.3.

4. The fourth and the final layer is composed of peers of the blockchain's P2P network. This is where the transactional history (of both data related to a particular use case—such as access control or provenance data—and the execution state history of the smart contract layer) is stored in different blocks of the chain. These nodes maintain the integrity of the chain through distributed storage and reach consensus on the version of the blockhain that is stored in a distributed manner among these nodes.

In Fig. 3.1 it can further be seen that there are two more aspects of the overall framework dealing with namely:

Governance by blockchain: This portion of the framework deals with automating the relevant policies and rules depending upon a use case at hand. The relevant rules and policies related to a use case are implemented by the USCs sublayer. The ramification (legal or technical) can be resolved through OSC by issuing a relevant PEI to the relevant USC. Further, a PEI maybe triggered by an ML-based monitor. The input to this ML-based monitor can be in the form of the transactional data produced by the use case at hand and stored on the blockchain or traffic flow data of the P2P network as analyzed in [12] and [6]. The main task here would be to create an analytical model of both USCs and OSC's state machines along with analysing their state transition function Γ.

Governance of blockchain: This portion of the framework manages the protocols and decisions related to the underlying blockchain system. These could include the block size or which consensus protocol to use. An informed decision could be made by a policy-making body in an iterative manner. Each iteration can consider the feedback from the underlying P2P network. This feedback can be in the form of smart contract execution data stored on blockchain. This can lead to a more systematic approach of development i.e., a system prototype that can be tested and improved in an iterative manner, like described above, before deploying it into the real world.

In summary, I would want to adopt such an approach for my doctoral investigation where I do not necessarily have to reinvent the wheel. Instead, I am more interested in the careful and intelligent rearrangement of the existing well established techniques and tools such as ML, smart contracts and the blockchain itself. The inspiration to adopt such an approach comes from the observation that the blockchain itself does not propose new tools and techniques rather it is merely a clever rearrangement of the existing techniques such as cryptography and hashing.

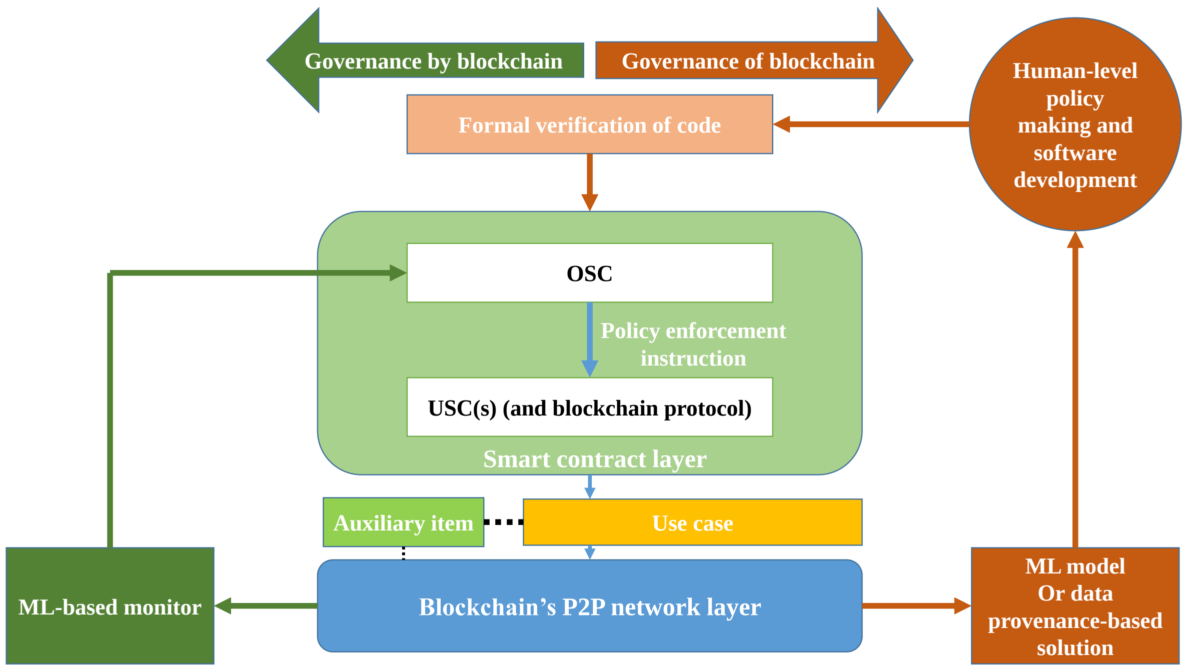
Governance by blockchain
Governance of blockchain
Human-level policy making and software development
Formal verification of code
OSC
Policy enforcement instruction
USC(s) (and blockchain protocol)
Smart contract layer
Auxiliary item
Use case
ML-based monitor
Blockchain's P2P network layer
ML model Or data provenance-based solution

Figure 3.1: Proposed governance framework based on layered design paradigm

3.3 Experimental setup

3.3.1 Software tools for experimentation

So far I have made myself familiar with the following experimentation tools.

- Truffle[1]: I have also tried a sample blockchain application of a pet-adoption shop with a web UI. Right now this seems to be the most lucrative tool and method to kick start the experimentation.
- Online solidity compiler[2]
- Method from the FC_16 paper [8]: this is based on using python implemention of Ethereum locally with smart contract coding in Ethereum project's serpent language. I still have to try this properly.
- Ethereum's testnet through its wallet: You can write a smart contract in the interface provided in the Ethereum' wallet and test onto the testnet network for development purposes.

[1]http://truffleframework.com/tutorials/pet-shop
[2]https://remix.ethereum.org/#version=soljson-v0.4.14+commit.c2215d46.js

3.3.2 Experimental setup

Here I will explain in detail all the necessary ingredients required for this experimental setup and relationship among them.

TestRPC

TestRPC is an Ethereum node. This is a fast way to get a test network up and running to test and deploy the smart contracts locally. When TestRPC starts it creates a few accounts with addresses.

Decentralised applications (Dapps)

Decentralised applications or as they are commonly referred to in blockchain/smart contracts community as Dapps are applications built on top of (usually) Ethreum's blockchain. These Dapps can be interacted with using *web3.js*[3] JaveScript application programming interface (API) for the purposes of building a web user interface for one's Dapps.

MetaMask[4]: acts as a bridge between a web browser and a Dapp developed on top of Ethereum's blockchain. One of the distinguishing features of this is that it allows one to interact with a Dapp without running an actual Ethereum client.

Dapp development frameworks

I have started my experimentation with Truffle[5]. It is one of the Dapp development frameworks that allows one to compile and deploy (migrate in truffle's jargon) to a blockchain network (in my case right now I deploy my contracts to TestRPC).

3.3.3 Experimental evaluation approach

I am considering following ways to conduct my experimental evaluation.

Synthetic and trace driven simulation: I will start off my experimentation with a model driven simulation. This work will be similar to the one as evaluated in the case of ProvChain [16] for data provenance use case of blockchain. OwnCloud[6] was used to simulate cloud storage with Chainpoint[7] as the underlying blockchian that generates a receipt for each data entry. I am also considering a tool similar to

[3]`https://github.com/ethereum/web3.js/`
[4]`https://metamask.io/`
[5]`http://truffleframework.com`
[6]`https://owncloud.org/`
[7]`https://chainpoint.org/`

ownCloud but with Truffle's framework to write smart contracts and interact with a blockchain and generate receipts for data entries. I can use a similar technique of using hooks, as in ProvChain, to detect operations performed on a piece of data that can further trigger specific functions of related smart contract(s). I am considering to start off my experimentation by implementing a simple use of arbitrary data (variable) storage example on blockchain[8] and tracking arbitrary changes to it.

Deployment in the wild: This will be the most challenging step. I will consider lessons learned from the above two steps. I take motivation from the work of [9] where the authors studied the relationship between two cloud providers. Given the mixed nature (as in involving technology and law) of my problem space in terms of stakeholders this will be quite a subtle challenge. However, after refining my code by including insights from lessons learned, Truffle provides the provision to migrate a smart contract to the actual blockchain e.g., of Ethereum with an interface to real-world stakeholders such as real-world cloud providers. Ultimately, as explained above in Section 3.3.2, I envision a web-based user interface that provides a control panel showing a user her choices such as applications she has signed up for and the related permissions with inforamtion realted to data provenance. Such a web-based tool will be interfaced with an underlying blockchain-powered set of smart contracts.

[8]`http://truffleframework.com/tutorials/debugging-a-smart-contract`

Chapter 4

Plan and timeline

The summarized version of the following discussion with the corresponding timetable for milestones can be seen in Table 4.1. In Table 4.2 I highlight a few venues that can be chosen to submit my research results and insights. All of these venues (except for NSDI and SIGCOMM which are more generic), as of 2017's call for papers, contained a research interest around blockchain theory and application.

4.1 Second year's research plan

4.1.1 First quarter

I intend to extend my literature review and prerequisite knowledge of the field and the use cases. This literature review will extensively look at different use cases of data ownership and provenance. I am interested in studying the implications that may arise after blockchain in implemented for a particular use case. Other purpose is to compare the exiting state-of-the-art techniques that try to solve the issues such as scalability, regulation and data ownership and provenance in blockchain-based ecosystems. The work to write a survey paper based on this plan is already underway. This can be consider as the *proof-of-work* for my PhD's first year.

4.1.2 Second quarter

Aspect: Governance by blockchain: I am going to use the experimental setup as described in Section 3.3.2 to implement two use cases of data ownership (Section 1.2) and data provenance (Section 1.3).

- **Data ownership use case** experimental setup. It will be similar to the one as described in [25] and discussed in Section 1.2. Specifically I will look at the problem of reconciling the third parties with the owners of data under a legal policy framework. I will also be evaluating the overhead caused in terms of time and storage by blockchain as an access control manager.

- **Data provenance use case** experimental setup based on the discussion in Section 1.3 and 3.3.3. Specifically I will look at the audit, accountable cloud computing, and forensics aspects. Again, I will be interested in evaluating how much overhead is caused by introducing blockchain-based data provenance for cloud computing similar to the one as discussed in [16].
- I am also interested in exploring the common ground and a link between two experiments as described above for an overall *ownership preserving data provenance system.*
- The most important outcome is to document all the insights collected and lessons learned while studying the above two aspects.
- As the final outcome of this phase of experimentation, my goal is to come up with an effective overarching smart contract (OSC that will effective govern/automate the use cases of data ownership and data provenance). This will include the analytical model in terms of state machine design for the OSCs and their associated state transition functions Γ_{own} for data ownership and Γ_{prov} for data provenance and implementation of their smart contracts. I hope to study these two use cases together for an overall accountable system. The overall OSC for such a system will ideally be capable of resolving any logical ramifications that might arise while such use cases are executing on top of blockchain hence implementing an effective and automated governance by blockchain. In order to study different legal and logical ramifications I intend to adopt a similar approach as discussed in [9].
- Also, as shown in Fig. 3.1, I will also be interested in looking at any *auxiliary items* that might be needed in order to fully automate a use case as for example off-blockchain storage solutions to store provenance (as in [16]) or user data (as in [25]).

4.1.3 Third quarter

Aspect: Governance of blockchain: This phase will be a comparative study, for example, of different consensus protocols that can be used for a blockchain-based system.

- One of the aims of this phase is to look at the potential of implementing a *formal verification of code* layer as used by Tezos [10] and as shown in Fig. 3.1.

- The part of OSC will be written here that will reconcile the community of blockchain stakeholders for a use case and the update to the system in general.
- I will also look at the potential of using a *ML-based monitor* so that a more informed updates can be issued to upgrade the system.

4.1.4 Fourth quarter

This phase will be dedicated to document my experience in the above three quarters in terms of lessons learned and insights gained while looking at the aspect of *governance by blockchain*. I will be looking at how effectively my OSC is performing in case something malicious (in phase I) or controversial (specially in phase II of experimentation) happens during the execution of USCs layer that automates a use case. I am planning to write my first draft of final thesis in this phase as well.

4.2 Third year's research plan

4.2.1 First half

During the first half of my final year of PhD I intend to conduct an extensive third phase of experimentation. The aim is to harmonize the phases I and II of experimentation as described above. The aim is to implement a holistic system that implements governance of and by the blockchain as shown in Fig. 3.1 for an overall *ownership preserving data provenance system*. I will also highlight the potential for future work that can be carried out either by myself or anyone else that finds herself interested in my topic.

4.2.2 Second half

This half will be dedicated to write a quality doctoral thesis. Ideally, after the acceptance of an article in the above phase I will incorporate the peer review, Jon's and my examination committee's comments to produce the final version of my thesis draft.

Table 4.1: Summary of Research plan and timetable

Plan	**Output and milestones**
Second year: First quarter (Feb 18-Apr 18)	
Extending the theory and literature review	i) Submission to IEEE Communications Surveys and Tutorials. ii) Post viva, refinement of my research problem space.
Second year: Second quarter (May 18-Jul 18)	
Experimentation phase I	OSC smart contract implementation for data ownership and data provenance use cases
Second year: Third quarter (Aug 18-Oct 18)	
Experimentation phase II	Implementing the *governance by blockchain* part of the framework as shown in Fig. 3.1.
End of second year: Fourth quarter (Nov 18-Jan 19)	
Documentation	i) Result submission to a reputable venue for publication. ii) First draft of my dissertation with the focus on lessons learned. iii) Second year's report, which is due on **January 31, 2019**.
Third year: First half (Feb 19-Jul 19)	
Experimentation phase III	i) Harmonize experimentation phase I and II. ii) Automate an overall governance framework as shown in Fig. 3.1. iii) Document the insights from the results and submit for a publication at a reputable venue. iv) Documentation focus: lessons learned and highlight the potential for future work.
Third year: Second half (Aug 19-Jan 20)	
Thesis writing	Submit the final version of my doctoral thesis for evaluation.

Table 4.2: Possible venues for the submission of research work

Venues	**Times**
Financial cryptography and data security	**Submission:** November; **Conference:** April
IEEE European Symposium on Security and Privacy	**Submission:** August; **Conference:** April
IEEE International symposium on Programming and Systems	**Submission:** December; **Conference:** April
IEEE/IFIP Network Operations and management Symposium	**Submission:** January; **Conference:** April
IEEE INFOCOM	**Submission:** July; **Conference:** April
IEEE International Conference on Cloud Computing (IEEE CLOUD)	**Submission:** January; **Conference:** July
IEEE Communications Standards	**Submission:** January; **Publication:** September
IEEE International Conference on Computer Communications and Networks (ICCCN)	**Submission:** March; **Conference:** July
IEEE/ICACT2018	**Submission:** December; **Conference:** February
NSDI	**Submission:** September; **Conference:** April
ACM SIGCOMM	**Submission:** January; **Conference:** August

Bibliography

[1] Blockchain accelerates insurance transformation, accessed 08-february-2017.

[2] The guardian view on the digital economy bill: a last chance to get it right, accessed 08-february-2017.

[3] Hyperledger Architecture, Volume 1. `https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf`, Note = [Accessed online: August 01, 2017].

[4] Ali, M., Nelson, J. C., Shea, R., and Freedman, M. J. Blockstack: A global naming and storage system secured by blockchains. In *USENIX Annual Technical Conference* (2016), pp. 181–194.

[5] Bacon, J., Michels, J. D., Millard, C., and Singh, J. Blockchain demystified.

[6] Biryukov, A., Khovratovich, D., and Pustogarov, I. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 15–29.

[7] Carata, L., Akoush, S., Balakrishnan, N., Bytheway, T., Sohan, R., Seltzer, M., and Hopper, A. A primer on provenance. *Communications of the ACM 57*, 5 (2014), 52–60.

[8] Delmolino, K., Arnett, M., Kosba, A., Miller, A., and Shi, E. Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. In *International Conference on Financial Cryptography and Data Security* (2016), Springer, pp. 79–94.

[9] Dong, C., Wang, Y., Aldweesh, A., McCorry, P., and van Moorsel, A. Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing. *arXiv preprint arXiv:1708.01171* (2017).

[10] Goodman, L. Tezosa self-amending crypto-ledger white paper, 2014.

[11] Kosba, A., Miller, A., Shi, E., Wen, Z., and Papamanthou, C. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *Security and Privacy (SP), 2016 IEEE Symposium on* (2016), IEEE, pp. 839–858.

[12] Koshy, P., Koshy, D., and McDaniel, P. An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security* (2014), Springer, pp. 469–485.

[13] Labs, P. Filecoin: A Decentralized Storage Network. `https://filecoin.io/filecoin.pdf`, 2017. [Accessed online: January 09, 2018].

[14] Lamport, L. The part-time parliament. *ACM Transactions on Computer Systems (TOCS) 16*, 2 (1998), 133–169.

[15] Lamport, L., et al. Paxos made simple. *ACM Sigact News 32*, 4 (2001), 18–25.

[16] Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., and Njilla, L. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (2017), IEEE Press, pp. 468–477.

[17] Merkle, R. C. Protocols for public key cryptosystems. In *Security and Privacy, 1980 IEEE Symposium on* (1980), IEEE, pp. 122–122.

[18] Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. `https://bitcoin.org/bitcoin.pdf`, 2009. [Accessed online: August 01, 2017].

[19] Pasquier, T., Singh, J., Powles, J., Eyers, D., Seltzer, M., and Bacon, J. Data provenance to audit compliance with privacy policy in the internet of things. *Personal and Ubiquitous Computing* (2017), 1–12.

[20] Sasson, E. B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., and Virza, M. Zerocash: Decentralized anonymous payments from bitcoin. In *Security and Privacy (SP), 2014 IEEE Symposium on* (2014), IEEE, pp. 459–474.

[21] Singh, J., Pasquier, T., Bacon, J., Powles, J., Diaconu, R., and Eyers, D. Big ideas paper: Policy-driven middleware for a legally-compliant internet of things.

[22] Singh, J., Powles, J., Pasquier, T., and Bacon, J. Data flow management and compliance in cloud computing. *IEEE Cloud Computing 2*, 4 (2015), 24–32.

[23] Walport, M. Distributed ledger technology: Beyond blockchain. *UK Government Office for Science* (2016).

[24] Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper 151* (2014).

[25] Zyskind, G., Nathan, O., et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE* (2015), IEEE, pp. 180–184.