The Blockchain: A Comparison of Platforms and Their Uses Beyond Bitcoin

Research · February 2017

DOI: 10.13140/RG.2.2.23274.52164

3 authors, including:

Lisa Liu-Thorrold
UNSW Sydney
3 PUBLICATIONS **24** CITATIONS

SEE PROFILE

The Blockchain: A Comparison of Platforms and Their Uses Beyond Bitcoin

M. Macdonald, L. Liu-Thorrold, R. Julien
COMS4507 - Advanced Computer and Network Security
The University of Queensland
{*millie.macdonald, lisa.liu.thorrold, romain.julien*}*@uq.net.au*

***Abstract*—This paper presents a comparison of five general-use blockchain platforms. We first discuss how the blockchain is used in Bitcoin, before looking at how blockchain technology can be used in applications outside of Bitcoin. We conduct an investigation into the blockchain platforms available for users to build custom applications on top of. We conclude with a summary of our findings and some suggestions for future work.**

1. Introduction to Blockchain

After capturing worldwide attention through its novel combination of ideas, underpinned by decades of research, Bitcoin has become one of the most successful digital currencies to date [1]. Blockchain technology was key to this success, and although conceived to support the ideas and goals of Bitcoin, researchers and esoteric thinkers alike have since realised the potential of blockchain in applications other than Bitcoin.

This potential is because the blockchain was designed to support the unique requirements of a digital currency, which can be desirable in many other situations. These requirements include decentralised management, anonymity, immutability, forgery resistance, and prevention of the double spend problem - a non-trivial problem to solve in a distributed setting that does not have an overseeing centralised entity.

Since the release of the white paper by Satoshi Nakamoto proposing Bitcoin as a viable peer-to-peer digital currency, academic research in this space has increased and other applications using blockchain technology have begun to surface. Some examples include reputation systems [2], contract management systems [3], and digital content distribution systems [4]. In response, several blockchain platforms have been developed. These platforms allow users to create their own applications above the blockchain stack and leverage the benefits of the blockchain.

To our knowledge, there is currently no literature available that investigates existing blockchain platforms and their suitability for developing applications. Thus, we have performed an analysis of five general-purpose blockchain platforms, highlighting their defining features and limitations. We aim to provide an in-depth comparison to help developers select a blockchain platform for their application.

The rest of this paper is organised as follows. In section 2, we provide a technical overview of blockchain in the context of Bitcoin. Section 3 demonstrates how blockchain technology can be applied to different applications using case studies. In section 4, we describe our methodology for comparing various blockchain platforms, including a brief explanation of the features we focused on. Section 5 presents the results of our investigation, and we conclude and describe future work in section 7.

Included in the appendix is a brief report for the implementation component of the assignment - whereby a simple, proof of concept reputation application was created using the Ethereum platform. It gives a description of the application, as well as providing an overall evaluation of the Ethereum platform with respect to our application. Potential improvements and future work are also suggested.

2. Blockchain in Bitcoin - Technical Overview

2.1. Introduction

In this section we give a technical overview of blockchain, explained in the context of Bitcoin. Many of the details were obtained from the original Bitcoin whitepaper [1].

Bitcoin's blockchain is essentially a distributed ledger system that records transactions conducted in the Bitcoin network. Each transaction is characterised by one or more transaction inputs (previous transactions from which the user has received Bitcoins) and one or more transaction outputs (users to send the Bitcoins to). Bitcoins are transferred by the current owner signing a transaction which transfers value from the inputs to a new owner(s) as identified by the outputs, which are Bitcoin addresses. The new owner then repeats this process to transfer the coins to the next user.

Bitcoins can thus be visualised as a chain of digital signatures, whose ownership and validity can be verified at every step along the way by traversing the public ledger and tracking the coin back to its creation [1].

2.2. Verification

Once the owner of the coin has broadcasted their transaction into the peer-to-peer network, it must undergo a

verification process called "mining" before it can become a part of the public ledger. For efficiency, these transactions are grouped into blocks for verification - hence the name blockchain. Verification is then performed by miners who devote computing power to solving a puzzle. This is the "proof of work" mechanism. In Bitcoin, this involves finding a value that when hashed with an algorithm such as SHA-256, the resulting hash begins with a certain number of zero bits.

The number of zero bits is determined by the system and is self-adjusting, as the puzzle's difficulty is exponential to the number of zero bits required. This ensures that the time to perform the proof of work for a block averages at 10 minutes despite fluctuations in the number of mining nodes and the increasing computational capabilities of the miners as hardware improves over time.

Verification is a competitive process as the first miner to solve the puzzle gets the right to broadcast the new block through the network, and wins the mining incentive (explained below). Blocks are only accepted by other nodes in the network if the transactions are valid and transaction inputs have not already been spent, which addresses the double spend problem. Acceptance of the broadcasted block is acknowledged by the rest of the nodes in the network by abandoning their attempt at solving the proof of work puzzle for the current block, and moving on to create the next block in the chain [1].

Because of the design of the proof of work mechanism, nodes with higher resource capacity have a higher probability of solving the puzzle first. Consensus on the integrity and balance of the blockchain is represented by the longest chain in the network, which has the most resource investment by proof-of-work.

2.3. Incentive Mechanism

Currently, the miner of the accepted block is rewarded 25 Bitcoins. This serves as an incentive to get people to mine blocks, and is the only way to generate new Bitcoins. This is in contrast to a centralised approach where a central authority is able to issue money to put into circulation.

The incentive mechanism is also designed to help prevent attacks against the blockchain. To safeguard against forgery, blockchain is by design practically immutable. However, if an attacker does wish to modify the contents of a previous block they would need to redo the proof of work for that block and all following blocks, up to the current block. The incentive mechanism encourages attackers that have such computing power to instead use it to mine Bitcoin legitimately.

An interesting feature of Bitcoin is that the reward for mining a block is halved every 210,000 blocks. At 10 minutes a block, that's approximately every four years. The reward per block mined started at 50 Bitcoins when Bitcoin was first introduced, and will decrease to 12.5 Bitcoins later this year. The amount of Bitcoin in the system will converge to a total of 21,000,000 Bitcoins which is expected to be around 2140 [5].

Note that the value of Bitcoin has had a volatile history, ranging from as little as less than USD $0.1c to as much as USD $1132 over the past 8 years [6]. The current reward of 25 Bitcoins is, at the time of writing, equal to approximately AUD $16,000.

2.4. Double Spend and Decentralisation

Attempts by the owner of a Bitcoin to double spend a coin by issuing two transactions for the same coin are thwarted via the verification process. If a node completes a proof of work puzzle on a block that contains a coin that has already been spent, it will be rejected by the rest of the nodes in the network, and subsequently will not be added to the blockchain.

This does however depend on the majority of the nodes being honest, which is encouraged by the incentive mechanism as discussed above. Furthermore, the decentralised nature of Bitcoin creates a reliable system which, unlike a system with an overseeing centralised entity, does not have a single point of failure. Finally, its transparency allows the authenticity of a coin to be ascertained on inspection, and coins to be traced along the blockchain back to their creation.

2.5. Scalability

Much discussion currently surrounds Bitcoin's block size limit, which restricts blocks to 1Mb of transaction and header data. As blocks are mined every 10 minutes this limits the number of transactions per second (TPS) to a theoretical limit of 7 TPS [7]. As Blockchain became more popular and more nodes joined the network, the number of transactions increased and this limit became a significant problem. If the transaction creation rate increases too much it could surpass the rate at which transactions are added to the blockchain, creating a backlog of transactions.

Several solutions to this have been proposed (e.g. increasing the block size limit) however as the community is currently unable to agree on the best method, nothing has been implemented yet.

3. Case Studies: Uses of Blockchain Technology

In this section we provide case studies of applications using blockchain technology in order to demonstrate its potential beyond Bitcoin. The case studies include a reputation system [2], a smart contract system [3] and a digital content distribution [4].

3.1. The Blockchain-Based Digital Content Distribution System

The appropriate distribution of digital content with commercial value is a challenging problem. Access to digital content is controlled by Digital Rights Management (DRM)

technology, and is generally enforced through hardware and software to preserve the content providers' rights and revenues.

The typical procedure for a user accessing commercially digital content is as follows [9]:

1) The digital content is encrypted. Users who want to access this encrypted content send a request to the DRM technology provider through the Internet for the license.
2) The DRM technology provider determines the usage policies based on the request. A financial transaction may be conducted in exchange for the issuing of a license.
3) The license is packaged and transferred to the client through the Internet so the content can be decrypted and used on the user's device that the request originated from.

Based on this procedure, there are a couple benefits that a blockchain implementation would provide over a centralised approach. The main benefit of using a blockchain implementation is that content author can control the distribution of their content themselves, without the need for a third-party DRM provider.

There are two obvious benefits of this. The first is that the author's profits would likely increase as DRM providers usually take a financial commission for each license sold. The second benefit is improved security. With the traditional system, employees of DRM technology providers can undermine the content provider's rights by distributing digital content to people who are not authorised to use the content. As a blockchain-based system removes the need for a DRM provider, it also removes this risk.

The authors of [4] demonstrated the viability of a blockchain-based digital content distribution system even for files of significant sizes (e.g. ultra-high resolution video content). Their prototype utilised video content that was compressed using High Efficiency Video Encoding (HEVC). Key management was handled by the modification of the header of the resulting compressed data each time, with only the holder of the correct secret key having the ability to use the digital content meaningfully. The implementation proposed by the authors contains three key components:

1) The content owner has two main responsibilities: to upload the digital content, and to control the rights of the content that they are distributing. The rights of the digital content are controlled by modifying metadata that has been added to the content.
2) The user who would like to access the digital content is required to run two applications: the licensing control application, and the player of the digital content. The user first conducts a transaction for the license of the content which gets recorded on the blockchain. The license control application retrieves the rights information of the content from the blockchain. The license control application controls the content player based on the retrieved license and the user's rights.
3) The mining server generates new blocks which include rights information for the digital content. The proof of work mechanism is very similar to that of Bitcoin. For the demonstration, the authors limited the mining time to 10 seconds, and hardcoded the number of zeros required at the start of the hash to be 4. Realistically, in a production environment this will need to be increased to guarantee the integrity of the system.

While the authors have not decided on an incentive mechanism for mining, they have considered paying Bitcoins to the miner. Payment to the miner in the form of some credit which can be accrued and exchanged for digital content licenses is also possible.

3.2. Blockchain Contract: A Complete Consensus Using Blockchain

Contractual documents are one method by which people can establish consensus on a topic. One example of this is a purchase and sale agreement when a buyer agrees to buy an item that a seller agrees to sell. Consensus is represented by a signature from the participants in the contractual agreement and management of these are generally left to the individuals. In a heavily contract based society, management of these can be difficult, especially as more and more are made each day, whether the user is consciously aware of making them or not. Furthermore, these contracts can be vulnerable to modifications which may not be authorised by all parties in the agreement, and difficult to track and verify as time passes.

Blockchain represents a sensible solution to address these problems due to its immutability by strong resistance to attack, and public availability. It does however require some modifications to make it suitable for a multi-way transaction as all participating parties must explicitly agree to the contract (and not just the sender as in Bitcoin).

The authors of [3] propose the following modifications to make the blockchain technology that is used in Bitcoin suitable for managing contractual documents:

1) A transaction, which represents consent to the contract by the sender, contains the following fields:
   - Destination address
   - The hash value of the previous transaction
   - Information about the sender including the e-signature of the sender and the hash value of the sender's address
   - The details of the contract
2) The participants create a chain of transactions where the first participant creates a transaction and sends it to the next participant in the contract. This is continued until all participants have received, signed and sent the transaction to the next participant in the chain. When the last participant has received and signed the transaction, it is sent back

to the first participant who confirms the contract contained in the transaction and forwards it to a third party mediator who records the trail of consensus and adds this to the blockchain.
3) To protect the contents of the contract, the contract is stored in encrypted form on the blockchain, so that only those who have a corresponding decryption key can decode the contract.

The authors of this paper do not mention proof of work and incentive mechanisms. However one possibility would be for the incentive mechanism to be funded by "establishment fees" paid by contract participants.

3.3. Rep on the Block: A Next Generation Reputation System Based on the Blockchain

Reputation systems are one way to express how trustworthy a person or entity is to society. They are most commonly used on e-commerce websites where financial transactions require the buyer to have some level of trust in the seller. Some problems that exist in current reputation systems include:

1) The reputation calculation algorithms are often centralised, and closed source. In this case, users often do not understand how reputation is calculated. In comparison, transparency increases a user's trust in the system by making it easier for them to understand how an individual or entity gained their reputation.
2) People are more likely to conduct transactions with the sellers or service providers with the most favourable reputation. It's also human nature for people to be conscious of how society perceives them. It's therefore in the seller or service provider's interest to ensure that their reputation reflects them in the most positive light, even if it's not a reflection of reality. This unfortunately makes reputation systems targets for gaming. For example, sellers may be able to forge multiple identities to boost the reputation of their principal account, and simultaneously commit acts of sabotage to damage the reputation of competitors (sybil attack). Gaming can also be done in collusion with multiple people to boost each other's reputation.

The approach taken by [2] proposes a blockchain based reputation system designed to withstand such attacks. To do this, the authors outline the following key features of their system:

1) Feedback from the transaction is represented by a single binary digit which signifies whether or not the user received the item that was requested. The item is signed by the sender's private key and is sent to the requesting user.
2) Once the correct item is received, the user sends a transaction consisting of the reputation score, a timestamp and a hash of the received item. This data is then encrypted with the receiver's private key and sent to the miners. This proposes to solve the problem where the reputation could be updated without a valid transaction, as there now exists cryptographic proof on the occurrence of a transaction.
3) To prove that a sender has sent a file, and the receiver has received a file, transactions are validated by the miner who contacts each user involved in the transaction and requests a signed proof which contains the file hash and a random nonce sent by the miner. Validated transactions are collated into blocks before performing the proof of work using the same method as Bitcoin.
4) To deter users from creating multiple identities within the system, the authors make joining the network expensive, as well as limiting a user's account to one per IP address. Although this does not prevent an attacker from launching a sybil attack on the system, this does restrict the attack to more powerful adversaries. The system also does not take into account multiple transactions between the same users when calculating the reputation score which makes a collusion attack more difficult to achieve.
5) To solve the problem of gaining trust from new users who initially have no reputation, these new users must stake a small amount of Bitcoin into a holding pool before they are allowed to sell goods or services. This stake is returned to the seller if the transaction was successful, otherwise it is moved into a separate holding pool that is used to fund the incentive mechanism for miners in the network.

While the paper proposes a reputation based system that solves a number of issues present in current reputation systems, it still requires work to transform it into a deployable system. One example would be making the feedback from a transaction more reflective of a real world transaction (which is generally multifaceted) instead of representing it with a single binary digit.

4. Methodology for the Comparisons

A comparison was made between a selection of blockchain platforms using several criteria related to usability, flexibility, performance and potential. A short explanation of each criteria used to compare the blockchain platforms is included below.

Although probably not a major concern to most users, originally whether a platform was open or closed source was also considered. However all the platforms investigated below are open source so this was not included as a section for each platform.

We consider the platforms being open source to be conducive to adoption and innovation, as anyone who is interested can download a platform to use and/or modify. Small projects also benefit from being open source as there

are often more eyes looking at the code than if the project was closed source, which aids with the detection of errors or problems in code. Users can also expect a more responsive outcome to feature requests, or bug fixes, especially if the project is backed by a strong developer community.

4.1. Usability

The first point of comparison is how difficult learning and using a platform is. An indicator of this is the number of different methods available for interacting with the platform (e.g. APIs, GUIs, clients) and the level of platform-specific knowledge required to develop with a platform (e.g. does programming smart contracts require a platform-specific language). Platforms that required less specific knowledge were considered more favourably.

4.2. Support and Documentation

Another important aspect for the comparison are the quality and quantity of documentation and developer resources for each platform. These include explanations of a platform's design and features, technical implementation details, tutorials and examples. Such documentation is important for enabling users to get the most out of a platform. Generally, more documentation was considered better, especially if the documentation was aimed at a range of users (e.g. developers, miners, application users).

4.3. Development

Where possible the development history of a platform (e.g. from release notes or a Github repository's commit history) and the size of the community working to develop and maintain it was taken into consideration. A longer history general suggests a platform is better developed, and a bigger community gives an indication that development will continue.

4.4. Limitations and Flexibility

Although there are many blockchain platforms that are intended to be used only for certain tasks, this report focuses on the more general-purpose platforms. Flexible platforms which can be used for many purposes can be more useful, more profitable and are more likely to inspire innovation. Therefore, anything that could restrict or increase the variety of applications a platform could be used for was noted. Where possible, if and how a platform is being used currently (e.g. the number of different applications currently using a platform) was also investigated although this information was often hard to find.

4.5. Scalability

Where possible, an investigation was also made into how well a platform scales with network size and the number of transactions per second that need to be validated. As shown by the issues currently faced by Bitcoin, scalability is important to any blockchain platform that could be widely adopted.

4.6. Consensus and Incentive Mechanisms

As a central feature of a blockchain's design, the cost of consensus (e.g. computational power, energy or time), the difficulty of consensus (e.g. is it random or probabilistic?) and whether the difficulty can be changed were carefully considered. Whether an incentive was in place to encourage participation in the consensus decision was also studied, although not all consensus mechanisms require an incentive mechanism.

4.7. Currency

Also considered was the currency a platform uses in terms of what it is (e.g. real world currency, Bitcoins, something else) and how users obtain it (e.g. via mining or buying it with other currencies).

4.8. Security

Finally, the security of each blockchain platform, in terms of the security of transaction data, user anonymity and the security of the blockchain itself was considered. As blockchains are generally used to transact some value (monetary or otherwise) or data, the security of these transactions is a major concern to most users.

5. Blockchain Platforms

5.1. Ethereum

Ethereum [10] is a decentralized platform with a Turing-complete contracting language that allows the development of smart contracts. Smart contracts are applications that run on top of a custom built blockchain, similar to Bitcoin's. Ethereum's facility to develop smart contracts allows complex applications such as financial exchanges and insurance contracts to be executed on the distributed platform [11].

5.1.1. Usability. Ethereum clients [12] are analogous to virtual machines and allow developers to run Ethereum programs on their local machine. Having a client installed is necessary to access the blockchain and its smart contracts. It also allows the creation of transactions and the mining of blocks. Clients are mostly developed and founded by the Ethereum Foundation but there are also a few clients developed by a community of programmers. The most popular clients offered by the Ethereum foundations are go-ethereum (Go), cpp-ethereum (C++), and pyethapp (Python). A GUI interface called Mist is also available to interact with the Ethereum client, however Mist is not sufficient to do mining operations.

To facilitate the development of smart contracts, Ethereum also provides Mix, an IDE for the Solidity language. Solidity [13] is a programming language similar in syntax to JavaScript designed to compile code to run on the Ethereum VM. Support was originally planned for Serpent (a Python-like language) and LLL (a Lisp-like language) however Ethereum chose to focus on developing for Solidity due to time and money constraints. LLL has since been deprecated, though some clients still support Serpent. Various development frameworks such as Embark and Truffle have also been developed by the community to assist with development workflows.

5.1.2. Support and Documentation. Ethereum's website [10] provides links to resources for developers such as an FAQ, a dedicated Stack Exchange website [**?**] and extensive documentation for Ethereum Homestead (the latest version of Ethereum) including explanations, tutorials and examples. We also came across numerous tutorials and guides written by the community using a combination of development tools to get distributed applications up and running in minimal time. Overall, we consider the documentation to be useful and comprehensive.

5.1.3. Development. Ethereum was first announced in 2014, and both the Ethereum Foundation and community members have made much progress since [14]. The community in particular have authored extra clients for Ethereum, extending its functionality and reach [12]. Notably, Ethereum have a defined roadmap which both the Foundation and the community are working off. Because of this, Ethereum does not suffer from some of the problems Bitcoin does where extended discussion over how to do something stalls development.

5.1.4. Limitations and Flexibility. Ethereum has already been used to develop hundreds of blockchain-based apps of various purposes from financial applications to games [16], and it appears that Ethereum will remain popular for some time yet. Although it has some limitations (e.g. Ether is only available by mining, unlike Bitcoin), there are plans to improve several aspects of Ethereum's design such as the upcoming switch from proof of work to proof of stake (as explained below).

5.1.5. Scalability. Ethereum has not fully addressed the scalability issue seen in Bitcoin, however some of improvements have been made in terms of the block size limit which for Ethereum is a "block gas limit". "Gas" represents the number of computational steps required to execute a contract, with one unit of gas equating to one computational step. Each transaction in Ethereum contains a gas limit which is the maximum amount of gas that contract can use (i.e. the maximum number of computational steps executing a contract should take) [17] and at the time of writing each block can only contain a combined total of 4.7 million gas for all its transactions [18]. The block gas limit can also be increased or decreased by miners each time they mine a block by a factor of 1/1024 [19].

By making the block limit proportional to the computation required to execute the contracts, rather than the size of the data, and by making it dynamic, Ethereum have created a blockchain platform which automatically scales to deal with increased transaction numbers. This has already been used to increase the block gas limit from the original 3.14 million [20].

5.1.6. Consensus Mechanism. Ethereum currently uses a proof-of-work algorithm called Ethash, which is a memory-intensive consensus mechanism [21] [22]. As in Bitcoin, the difficulty automatically adjusts every block so that a block is created approximately every 15 seconds. Unlike Bitcoin however, Ethereum mining depends on a 1GB dataset which is generated from the headers of previous blocks every epoch - about every 30000 blocks or 5.2 days. Because this dataset can take a long time to generate, Ethereum clients need to generate and store future dataset in advance to prevent mining delay at the start of a new epoch.

Mining is done by choosing a subset from the dataset using the block's header and some nonce, and comparing a SHA3-256 hash of this data subset to some threshold. If the hash is less than the threshold, then the nonce is valid and the block is added to the blockchain.

In designing Ethash, Ethereum aimed to make an "ASIC resistant" consensus mechanism which reduced the benefit of joining mining pools, such as many miners do in Bitcoin [22]. Since Blockchains are intended to be decentralised networks, significant mining centralisation can introduce a risk to the network and users (e.g. 51% attacks). By making the consensus mechanism rely on memory rather than computational ability, Ethereum have made pool mining much less attractive. This also makes ASICs less useful, as top-range graphics cards are already quite capable for mining Ether. Thus anyone with ideas on how to improve memory bandwidth would be better off selling them to a graphics card company than trying to design an ASIC just for mining Ether [23].

There are plans to switch from the proof-of-work consensus mechanism to a proof of stake mechanism sometime this year. This algorithm is called Casper and requires nodes to place a security deposit to take part in the consensus process [24]. Consensus is reached by nodes repeatedly betting their deposit on which block to include in the blockchain next. Betting correctly earns a node their deposit back plus transaction fees, betting incorrectly means a node only regains part of their deposit. Because the "correct" bet is decided by majority vote, nodes' votes will gradually converge as they try to vote with the other nodes so they do not lose money.

If a node at any point produces anything "invalid" (e.g. try to double-spend) they will lose both their deposit and the privilege of participating in the consensus process. This will hopefully deter attackers as they stand to lose money if detected.

5.1.7. Incentive Mechanism. The incentive mechanism in Ethereum involves a reward of 5 Ether for mining a block, an amount of Ether based on the cost of the gas expended in the block and the current gas price, and an extra reward per any "uncles" included in the block. Uncles are blocks from forks in the blockchain whose parents are ancestors of the current block, up to 6 blocks back along the chain. Rewards are given for including uncles because it decreases centralisation incentive (miners who hear about blocks late because of network propagation delay still get a reward), and it increases blockchain security by augmenting the work done on the main chain [25].

5.1.8. Currency. Ether is the cryptocurrency used on the Ethereum platform. It is used to reward miners for executing a contract's requested operations which encourages contract programmers to write quality code as heavy computation costs more [26]. Ethereum also allows programmers to create their own cryptocurrencies using tokens. Tokens are created using the same coin API used to create Ether, making them compatible with any wallets or other contracts.

5.1.9. Security. Data and contracts in Ethereum are encoded but not encrypted, and all data is completely public [27]. Therefore, if sending sensitive data through the Ethereum network the user should encrypt it locally before sending it. However, because the data is publicly accessible, care should still be taken and highly sensitive data such as passwords probably should not be sent via Ethereum. There is some work being done to make sharing passwords and secrets via Ethereum possible (e.g. through code obfuscation) although this has not been implemented yet.

In terms of attacks against the blockchain, Ethereum has many of the same weaknesses as the Bitcoin blockchain (e.g. weak against 51% attacks), though work is also being done to decrease the risk of these.

5.1.10. Summary. Ethereum has several advantages over Bitcoin, and not just because it enables smart contracts. With its extensive documentation, dynamic block size limit, tailor-made consensus mechanisms (both Ethash and Casper) and fairer incentive mechanism, Ethereum could someday take over the from Bitcoin as the most popular blockchain platform. However, Ethereum is still being developed, and therefore has bugs that need to be fixed and improvements yet to be implemented. So while Ethereum can be used as a blockchain platform in its current state, it should be even better in the future.

5.2. IBM Open Blockchain (OBC)

Earlier this year (2016), IBM open sourced their Open Blockchain (OBC) project as part of the Linux Foundation's new blockchain project [28]. IBM's code now forms a core part of that project as Hyperledger Fabric, however the discussion in the section will be about the OBC project. The focus of OBC is to allow organisations to work with enterprise level technology to create applications that solve common business solutions in a novel and progressive way.

One of IBM's goals with OBC is the improvement of business workflows by using blockchain as a platform to issue, trade, manage and service assets. It also proposes to automate business processes by deploying business rules as smart contracts on the blockchain that can be validated by all stakeholders in a trusted way [29]. Other use cases include a manufacturing supply chain, an asset depository, and a communication platform [30].

The traditional blockchain model as it is does not satisfy the requirements that make it appropriate for use in business critical applications. One key example of this the lack of support for confidential and private transactions. Consequently, OBC differs from the traditional blockchain model used by Bitcoin in that nodes require permission to join the network. This is achieved by having an authority on the network that is able to issue identities for entities to transact on the network, with these identities having distinct levels of permissioning associated with them.

5.2.1. Usability. The Application Programming Interface (API) offered by OBC includes REST AND JSON RPC, events, and a Software Development Kit (SDK) to allow custom applications to communicate with the network [29]. OPC also provides a set of Command Line Interfaces (CLIs) to administer and manage the blockchain network. Programs running on the ledger - code that is deployed, executed and validated on an OBC network - are called chaincodes. Currently, chaincodes can be written in Golang, with support for Java and Javascript planned for later on this year.

5.2.2. Support and Documentation. OBC's support documentation is available as a GitHub repository [28]. It includes a whitepaper, an FAQ, a draft of the OBC protocol specification and several tutorials on how to use OBC.

5.2.3. Development. The GitHub history for both the OBC and Hyperledger repositories go back to September last year, although it is likely that IBM were working on OBC before that [31]. IBM also welcome contributions from the community [28]. These factors, plus the backing of IBM and collaboration with the Linux Foundation, suggest that development of OBC will continue well into the future.

5.2.4. Limitations and Flexibility. As OBC is still being developed, many details of its design are unknown. Also, although IBM offers consultative services for clients wishing to create a blockchain application using OBC to deploy on its Bluemix cloud platform (Blockchain as a Service) [36] they have not released any information about their client base and no known applications using it exist. Therefore, we can not accurately report on any limitations of its design.

5.2.5. Scalability. OBC's Usage FAQ [32] says that

> The current performance goal for OBC is to achieve 100,000 transactions per second in a standard production environment of approximately 15 validating nodes running in close proximity.

No further details are given, for example in terms of how this would be achieved, and as only a proof of concept implementation exists, we are not able to properly judge OBC's performance at this time.

5.2.6. Consensus and Incentive Mechanisms. OBC supplies two different consensus algorithms: the Byzantine Fault Tolerance (BFT) algorithm, and an augmented version of it called SIEVE to make it more suitable for business applications. A research paper outlining this algorithm in detail is due to be released in the near future [33]. OBC allows users to deploy different consensus algorithms to suit their application by allowing the algorithms to be plugged into the OBC architecture.

The concept of mining whereby users participate in the transaction validation process by investment of resources for a financial reward appears to be absent from IBM's Open Blockchain whitepaper. Instead, it allows users with the appropriate permission levels validate transactions before they are added to the blockchain [29].

5.2.7. Currency. OBC does not have a native currency, however one can be created by developers for their application using chaincode [29].

5.2.8. Security. A large portion of the OBC protocol specification is dedicated to security issues, including pseudo-anonymity of users (ensuring transactions ca not be linked to users or to other transactions) using public keys and digital certificates, access control policies and network security [34]. It is obvious that a lot of research has been done on ensuring OBC is as secure as possible, although some of the proposed features are yet to be implemented (e.g. the replay attack resistance mechanism is not available in the current version of OBC).

5.2.9. Summary. IBM's Open Blockchain platform would be best suited for business-oriented applications intended for streamlining record-keeping processes and deploying smart contracts in a secure manner. Unfortunately, aside from a few sample applications that demonstrate its use in the IBM's OBC repository, the project is still reasonably young. It is little more than a proof of concept implementation at the moment and is still missing several key features so, although it has promise, much more needs to be done before it can be considered suitable for most applications.

5.3. Intel Sawtooth Lake

Intel's experimental contribution to the Hyperledger project, Sawtooth Lake, is designed to be a highly modular and versatile distributed ledger platform [49]. It is quite similar to Bitcoin with two main differences - its extensible transaction types and its unique consensus algorithm.

5.3.1. Usability. Although Sawtooth Lake comes with a default set of transaction families that can be used to implement a fully functional digital assets marketplace, Sawtooth Lake also allows the user to create their own "transaction families" via the Python API [49]. Intel identifies three main components of a distributed ledger - the ledger's data model, the transaction language used to modify the ledger and the consensus algorithm - and defines a transaction family as a combination of the data model and transaction language.

There is also a minimal web API that allows access to blocks, transactions and state storage as well as sending transactions to the network, and which encodes query responses using CBOR or JSON [50]. Much of the functionality of the Python API seems to be missing from the web API however, so it is unlikely that the platform could be used entirely via the web API.

5.3.2. Support and Documentation. The Sawtooth Lake documentation is quite extensive, including explanations of the platform's features and implementation, plus tutorials for developers and system administrators, and an FAQ [51].

5.3.3. Development. As an open source project run by Intel, Sawtooth Lake has a lot of potential. The Intel Ledger GitHub [52] only shows work by half a dozen people since March this year, however it is possible that Intel employees were working on Sawtooth Lake long before that. Its part in the Hyperledger Project is also considered a good sign that development will continue.

5.3.4. Limitations and Flexibility. Currently, the main limitation of Sawtooth Lake is that the PoET consensus mechanism has not been fully implemented, and as such it is not as secure as it should be. Once this is implemented, the reliance on a Trusted Execution Environment (TEE) such as Intel's Software Guard Extensions (SGX) introduces a third-party which users have to trust and which may have its own security issues and limitations. It also means that users will have to use SGX-compatible Intel processors. Intel does however offer Quorum for those who do not want to or ca not use PoET so this may not be a problem.

To the best of our knowledge, Sawtooth Lake has not been used to build any applications yet. It was only open sourced in March however, so this may be because developers have not had the time to produce release-worthy applications.

5.3.5. Scalability. The documentation makes only a few mentions of scalability, though the two consensus mechanisms were designed to cater for networks of different sizes and with different requirements.

5.3.6. Consensus and Incentive Mechanisms. A significant highlight of Sawtooth Lake is that it includes two consensus algorithms, each intended for use in different situations due to their different performance trade-offs.

The first consensus algorithm, called Quorum Voting, is based on traditional Byzantine Fault Tolerance (BFT) algorithms, but uses multiple rounds of explicit votes to achieve consensus [49] based on adaptations of BFT algorithms by blockchain platforms Ripple and Stellar [53]. The

documentation contains little more information about this mechanism, except that it is intended for use by applications that require immediate transaction finality.

The second algorithm, a less resource-intensive algorithm developed by Intel, is called the Proof of Elapsed Time (PoET) algorithm [49]. Like Bitcoin's Proof of Work algorithm it's a lottery protocol where one leader is elected from the population with some probability, who then broadcasts the new block to the network. Other nodes then implicitly vote to accept the block by adding it to their local copy of the blockchain. However unlike Bitcoin, which uses a cryptographic hash puzzle to elect the leader, PoET uses a trusted execution environment (TEE) such as Intel Software Guard Extensions (SGX) to randomly assign each participant a wait time for each transaction block and the participant with the shortest wait time becomes the leader.

This algorithm meets the criteria for a good consensus algorithm as identified by Intel - fairness of leader election distribution among population, probability of election is proportional to resources contributed (i.e. processors running TEEs) and easy verification via the TEE. The low cost of participation (i.e. only having to wait some time, rather than use computational power as in Bitcoin) requires less energy, scales to thousands of participants, runs efficiently on any Intel processor that supports SGX and should encourage participation and thereby increase the algorithm's robustness.

However, the current implementation of Sawtooth Lake only simulates the behaviour of PoET running in a TEE and as such the fairness of the generation of wait timers ca not be assured. Because of this, Sawtooth Lake should only be used for experimental purposes and especially not for security sensitive applications. It also imposes a limitation upon the applications of the platform once PoET has been fully implemented and requires the SGX TEE, as it will then only be able to be run on SGX-compatible Intel processors.

There is no mention of an incentive mechanism in Sawtooth Lake's documentation, but as being chosen as the next leader is probabilistic a reward would probably not be appropriate.

5.3.7. Currency. Due to the flexibility of Sawtooth Lake and its extensible transaction types, there is no one currency used by all applications built using this platform. Instead, users are able to define their own currency. The inbuilt transaction families include a token asset type but it is unrestricted (anyone can create tokens) and non-consumable (owning one token is the same as owning infinite tokens as they are not consumed when used in a transaction) so they are not useful as a true currency [53].

5.3.8. Security. As stated above, the PoET algorithm is currently not secure, so at this time Sawtooth Lake should not be used for security sensitive applications. However, once it is fully implemented, its security will depend on that of SGX. SGX is a set of instructions which allows application to run in sectioned-off areas of memory called enclaves. This aims to protect sensitive data and code from disclosure or tampering, both when stored and at runtime, without affecting other applications running on the same machine [54]. Unfortunately, since Intel first introduced SGX in 2013, several weaknesses have been found in its design. These include secure assertions (i.e. digital certificates that identify the hardware environment and enclave) which are retrieved from an Intel database via the Internet [55], vulnerability to sidechannel attacks [56] and problems with access to enclaves by anti-malware applications which could make them ideal for running malware undetected [57]. There is therefore the possibility of attackers compromising the system and, for example, increasing their chances of being chosen as the new leader by controlling wait time assignments.

5.3.9. Summary. Sawtooth Lake has the potential to be a very flexible blockchain platform, with its extensible transaction types and two consensus mechanisms. Its reliance on SGX may be a drawback, due to the current security issues, the future limitations of needing SGX to use the PoET consensus mechanism and any security issues with SGX itself, but at least developers have the option of using Quorum instead. Being open source and an Intel project, it will be interesting to see how Sawtooth Lake develops. However in its current state it is not suitable for many blockchain-based applications, particularly any that require good security.

5.4. BlockStream Sidechain Elements

Sidechain Elements is an interesting blockchain innovation [37]. It introduces "sidechains" which are essentially standalone blockchains which can be integrated into other blockchains. This allows for the transfer of assets between two blockchain networks.

It also introduces a degree of modularity using "elements". These are stand-alone features which can be arbitrarily combined and added to a sidechain. Elements currently under development include Confidential Transactions (where only those involved in the transaction can see the amount transferred) [38] and Signature Covers Value (which invalidates a transaction's signature when its inputs are spent, allowing for faster transaction validation) [39].

Possible uses for sidechains include the creation of blockchains with advanced smart contracts or advanced privacy features (e.g. true anonymity), or as the basis for applications that allow banks to control their own digital currency or companies to control customer reward schemes [40].

5.4.1. Usability. As Sidechain Elements is still in the alpha phase, it is not yet available for general consumption [40]. The source code to run the Alpha Sidechain Elements demo is available on Github [41] where developers are encouraged to experiment with this code, and modify to build applications of their own. This demo has minimal functionality, runs only on a test network, and can only be interacted with through command line instructions. While a proper API,

client and/or GUI is probably being planned, no information on this is available yet.

That said, Sidechain Element's modularity could make creating a custom sidechain for an application quite easy, although the elements need to be better developed before they will be useful in this way.

5.4.2. Support and Documentation. As Sidechain Elements is in the alpha phase not much documentation exists. The project's GitHub [42] contains a few pages, as does the Elements Project website [43], however the pages are short and some are obviously incomplete.

5.4.3. Development. Blockstream as a company first appeared in 2014, after a funding round that netted them $21 million [44]. The Elements Project GitHub however goes back to 2009, with over 200 contributors [45]. Such involvement from the community would suggest a bright future for Blockstream, though work on the project seems to have decreased since mid last year.

5.4.4. Limitations and Flexibility. The modular design of Sidechain Elements means it could be a very flexible and adaptable blockchain platform. However, as it is still in alpha stage, its flexibility and any limitations that might affect its usage are unknown.

One potential limitation is the need for the other blockchains it interacts with (e.g. Bitcoin) to allow the connection between blockchains, which they may not. As Sidechain Elements relies on the currencies of other blockchains, having no mechanism for mining new coins, it is entirely dependent on being able to integrate with other blockchains.

5.4.5. Scalability. Due to the lack of documentation, it is hard to determine how well Sidechain Elements would scale. However, as sidechains are likely to have less nodes than a full blockchain and due to design decisions such as using federated consensus, we believe Sidechain Elements would scale relatively well.

5.4.6. Consensus and Incentive Mechanisms. Sidechain Elements proposes the use of a centralised federated consensus' security model to confirm transactions. This is where only certain nodes act as functionaries and validate transactions for each other using tamper-resistant hardware boxes with a special software stack embedded [46]. The transactions are confirmed when a majority of the functionaries sign the transactions. This is considered by the Blockstream developers to be more secure than having only one machine validating transactions and allows for better control over who is validating transactions [37]. For example, a bank running a sidechain can specify that only their own machines can validate transactions, and not those of their customers.

As the consensus mechanism does not involve mining, Sidechain Elements does not have an incentive mechanism.

5.4.7. Currency. Blockstream does not have its own currency. Value comes from other sources, is conceptually represented by tokens and can represent anything of value that can be traded on the sidechain platform. This value could come from almost any source, such as vouchers, coupons, currencies, deposits, bonds, and shares [47] [48].

As an example, if some Bitcoins were sent to a sidechain, they would be sent to and held by a multisignature address on the Bitcoin blockchain. The keys required to unlock these Bitcoins would then be transferred to the sidechain where their value would be represented by some tokens. These tokens could be traded within the sidechain as normal, or could be converted back to Bitcoins by using the given Bitcoin address and keys to transfer the required value of Bitcoins from the sidechain's holding address to another Bitcoin address [37].

5.4.8. Security. Very little information is available on what security measures Blockstream plans to implement in Sidechain Elements. Their choice of consensus mechanism does however suggest they are considering what extra features corporate blockchains would require (e.g. better control of who can validate transactions), and some of the elements under development are security-related.

5.4.9. Summary. Sidechain Elements is still in a highly experimental phase with a fully-developed production-ready client yet to be completed. While Sidechain Elements provides an interesting and unique way for developers to create blockchain based applications, it may take some time before realistic and meaningful applications can be developed using this platform.

5.5. Eris

Eris is a blockchain platform aimed at enterprise applications. Their main selling point is their capabilities based permissions, which define who can transact on the network, create smart contracts, validate transactions and many others [58]. Eris is also a modular platform that uses Docker to allow various components to be switched in and out with other compatible components. The default components are the Ethereum VM, the Tendermint Socket Protocol for consensus, the mint-client for talking to Terndermint, Eris's key signing daemon and a Solidity compiler [59].

5.5.1. Usability. While the modularity of Eris may be helpful in customising the platform for a certain use, it also means there is more for the developers and users to learn.

Because Eris uses the Ethereum VM [59] it also requires knowledge of Solidity or Serpent, which are programming languages specific to the Ethereum VM [13]. They are based on Javascript and Python, two fairly well-known languages, but this might still present an entry barrier. Eris also currently requires developers to find their own key signing daemon [59], which requires knowledge in this domain.

Eris could be a good platform to use if a developer wants to make a simple application using the recommended

components. However, for developers who want to use other components or make their own, there could be quite a bit of work involved.

5.5.2. Support and Documentation. The Eris documentation is quite extensive, including both explanations of its features and the design decisions made during development, plus several tutorials on how to use it [60].

5.5.3. Development. The first blog post on Eris's website is from December 2014 [61], so assumedly work on it started sometime before then. Eris has also recently partnered with a few other organisations involved in developing blockchain-related technology [62] [63] [64]. This suggests development of Eris will continue strongly, although its progress does depend on the development of the other technologies it uses (e.g. Ethereum and Tendermint).

5.5.4. Limitations and Flexibility. The Ethereum VM can not, as far as we know, be switched out for another component at this point. It's likely that no other applications with the required functionality exists (e.g. being Turing complete). As the Ethereum VM requires the code it runs to be written in a custom bytecode language, the only languages developers can currently program Eris contracts in are Solidity or Serpent [13].

Also, Eris's key signing daemon is currently for development only, and must be switched out with the developer's choice of key management system. Eris are however looking at giving proxy-based access to various security and key management systems in the future [59].

That said, once better developed Eris's modularity could make creating custom blockchains with it easier for those without domain-specific knowledge. This would however require the current components to be better developed and probably more components to be made available (e.g. a working key signing daemon).

5.5.5. Scalability. As a modular system, Eris's scalability is dependent on that of its components.

5.5.6. Consensus and Incentive Mechanisms. Again, as a modular system, Eris's consensus mechanism depends on which component is used (e.g. Tendermint uses a variation of the Byzantine consensus algorithm [66]). The incentive mechanism is also dependent on the components used, although as Eris is a permissioned blockchain where only certain nodes have the job of validating transactions, whether or not a reward is appropriate depends on the application.

5.5.7. Currency. Eris is intended to be used more for business logic than monetary transactions. That is, it should be used as a process auditor to track when some process (in the form of a smart contract) was created, executed, etc. rather than as a transaction auditor [67]. Therefore, although the currency is dependent on how the platform is composed, it is not a focus of the platform.

5.5.8. Security. Eris's security depends heavily on the components a particular algorithm is made of. Currently, Eris's key signing daemon is a major security flaw. In the future, any components that could be used with Eris would need to be analysed for any security flaws. Because this would need to be done for every component and every combination thereof, guaranteeing a particular application developed with Eris is secure could require quite some work.

5.5.9. Summary. In summary, Eris has the potential to be a good platform for enterprises to create custom blockchains to build applications on top of, but it (and the third party components it relies on) needs to be developed further before an accurate assessment can be made. Its modularity could be good (e.g. for quickly throwing together a blockchain with certain features) or bad (e.g. involves third parties). Hopefully its many partnerships are a sign that development of Eris will continue.

6. Results of Comparison

Overall, we found the best blockchain platform is currently Ethereum, although the other four all have potential. All of the platforms also have their own advantages and disadvantages.

6.1. Usability

Ethereum and OBC are far in front of the other platforms in terms of usability. Both have multiple methods for interacting with the platform with even more planned for the near future. OBC's plans to allow for contracts to be programmed in Java and Javascript (two fairly common languages) put it slightly ahead of Ethereum which requires knowledge of Solidity or Serpent, which are specific to the Ethereum VM.

6.2. Support and Documentation

Ethereum has the most extensive documentation and the most support options of all the platforms we investigated. Sawtooth is a close second, followed by OBC (although their documentation is only on GitHub). Sidechains Elements and Eris both have decent documentation, although Sidechain Elements's is obviously incomplete and too much of Eris's documentation is devoted to explaining various aspects of blockchains rather than how their platform works or how to use it.

6.3. Development

Ethereum and OBC are both part of the Linux Foundation's Hyperledger Project, although OBC has the advantage of being run by an existing corporation (IBM). Sawtooth Lake also has that advantage (being an Intel project) however it appears to be a much newer project. Ethereum is the oldest project of the platforms we investigated.

6.4. Limitations and Flexibility

Eris and Sidechain Elements both have the potential to be highly flexible platforms due to their modularity (and Sidechain Elements because it interlinks with other blockchains), although both need further development before they become truly useful. Sawtooth Lake's extensible transaction types also open up more possibilites in terms of what the platform could be used for.

Ethereum does not have modularity or extensible types, but unlike the others its flexibility has already been proven by the hundreds of applications that have already been made with it.

Conversely, in terms of limitations, Sawtooth Lake and Eris are the worst because of features which have not been fully implemented yet. Sidechain Elements is still under development so it's hard to give a good assessment, whereas no major limitations were identified for Ethereum.

6.5. Scalability

Ethereum is the clear leader in terms of addressing the scalability issue. Their dynamic block gas limit allows miners to change the block limit as needed to adjust the TPS of the network. Sawtooth Lake has also been designed with two consensus mechanisms, which could help with scalability issues although how much is unknown. The scalability of Eris and Sidechain Elements depends on the components used to create an application's blockchain, and OBC's documentation had only a scalability goal and no mention of how it would be achieved.

6.6. Consensus and Incentive Mechanisms

Ethereum's memory-intensive, ASIC resistant proof of work algorithm Ethash has several advantages over Bitcoin's, and their planned switch to a proof of stake algorithm could improve the platform even more. Ethereum's incentive mechanism also rewards more users, encouraging users participate in the consensus mechanism.

OBC offers two consensus algorithms, although one is simply a BFT algorithm and the details of the second have not been released yet.

Sawtooth Lake also offers two algorithms, one of which is again a BFT algorithm while the second is their custom PoET algorithm. Although potentially better for several reasons, PoET must be executed in a TEE such as Intel's SGX which introduces several new problems and risks. PoET also has not been properly implemented yet, so currently any applications created using it are not secure.

Sidechain Elements and Eris both use permissioned consensus mechanisms to restrict who can validate transactions as they are aimed at enterprise applications. Sidechain Elements however also requires tamper-resistant hardware boxes with a special software stack, while Eris's consensus and incentive mechanisms depend on the components used to create the application's blockchain.

6.7. Currency

Ethereum gives a reward to successful miners, as in Bitcoin, though unsuccessful miners may also receive rewards for mining uncle blocks. It also has tokens, which can represent other values in the network. OBC allows developers to create their own currency, while Sawtooth Lake allows the creation of currencies and transaction types. Value in a Sidechain Elements blockchain is represented by tokens, although the value actually comes from ther sources so no value is actually created in the network. The currency of an Eris blockchain depends on how it is composed, although Eris's focus is on business logic rather than monetary transactions.

Therefore, all the platforms have their advantages and disadvantages in terms of their currency, so their suitability to an application depends on its requirements.

6.8. Security

Most of the platforms make only brief mentions of security, except OBC whose protocol specification contains a large section on security issues and measures. Ethereum and Sidechain Elements are both investigating potential security improvements although these have yet to be implemented. Sawtooth Lake is not secure at this time due to the PoET algorithm currently not requiring a TEE, and future security issues could arise due to problems with SGX's security. Finally, Eris's security depends on the components the blockchain is built from, and currently the default key signing daemon should not be used for application development.

6.9. Summary

Overall, of the platforms we investigated Ethereum was the best in terms of documentation and support, development and scalability. Although Ethereum did not have any features that particularly increased its flexibility, it also did not have any major limitations. It has also been around for longer than the other platforms, and has already been used to create fully-developed applications. Of the non-permissioned consensus mechanisms it it is better than Sawtooth Lake because Sawtooth Lake's is not fully implemented yet. It also has no known security issues, with some security improvements planned for the near future.

OBC and Sawtooth Lake are also decent blockchain platforms although they have their issues. Too much information about OBC is unknown to be able to give a complete assessment of it at this time, and Sawtooth Lake's main consensus algorithm is currently only half-implemented and is not yet secure.

Eris and Sidechain Elements both have potential as general-purpose blockchain platforms, particularly for enterprise applications that require permissioned consensus mechanisms. Unfortunately they are both too underdeveloped at this time to be used for developing blockchain-based applications.

Once these blockchain platforms have been developed further, a better analysis could be made of their specific strengths and weaknesses. It is also likely that more blockchain platforms could appear in the future, which could also then be included in such a comparison. Our work here also focused purely on general-purpose blockchain platforms, so future work could expand on this to include specific-purpose blockchain platforms.

In the end, although blockchain platforms have progressed greatly since the concept was first introduced as part of Bitcoin, it will be interesting to see how they continue to develop in the future.

7. Conclusions

This paper has discussed how the blockchain is used in Bitcoin, some potential uses of it outside of Bitcoin, then presented a comparison of five general-purpose blockchain platforms. This comparison was based on several criteria important to a platform's suitability to being used to build custom applications. We found that Ethereum is currently the most suitable platform, although the others all suffered in various ways from not yet having been developed as much as Ethereum. Therefore, it was suggested that another comparison be made sometime in the future, to give these platforms time to mature. It was also noted that although blockchain platforms have come a long way since Bitcoin appeared, they still have a lot of potential left.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", 2009. [Online] Available: https://bitcoin.org/bitcoin.pdf

[2] R. Dennis and G. Owen, "Rep on the block : A next generation reputation system based on the blockchain" in *The 10th International Conference for Internet Technology and Secured Transactions (ICITST-2015)*, pp. 131 –138. doi: 10.1109/ICITST.2015.7412073

[3] H. Watanabe et. al., "Blockchain Contract: A Complete Consensus using Blockchain" in *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*, 2015, pp. 577 –578. doi: 10.1109/GCCE.2015.7398721

[4] J. Kishigami et. al., "The Blockchain-based Digital Content Distribution System" in *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*, 2015, pp. 187 –190. doi: 10.1109/BDCloud.2015.60

[5] G. Hurlburt and I. Bojanova, "Bitcoin: Benefit or Curse?" in *IT Pro*, vol. 16, no. 3, 2014, pp. 10 –15. doi: 10.1109/MITP.2014.28

[6] Bitcoin Help (2015, Nov). *Bitcoin Price Chart with Historic Events* [Online] Available: https://bitcoinhelp.net/know/more/price-chart-history

[7] Bitcoin Wiki (2016). *Scalability FAQ* [Online] Available: https://en.Bitcoin.it/wiki/Scalability_FAQ

[8] Bitcoin Wiki (2016, Feb). *Scalability FAQ* [Online] Available: https://en.Bitcoin.it/wiki/Scalability_FAQ

[9] Y. He, L. Hui and S. Yiu, "Avoid Illegal Encrypted DRM Content Sharing with Non-transferable Re-encryption" in *2011 IEEE 13th International Conference on Communication Technology (ICCT)*, pp. 703 –708. doi: 10.1109/ICCT.2011.6157967

[10] Ethereum (2016). *Ethereum Project* [Online] Available: https://www.ethereum.org/

[11] S. Omohundro, "Cryptocurrencies, smart contracts, and artificial intelligence" in *AI Matters*, vol. 1, no. 2, 2014, pp. 19 –21. doi: 10.1145/2685328.2685334

[12] Ethereum (2016). *Choosing a Client* [Online] Available: http://www.ethdocs.org/en/latest/ethereum-clients/choosing-a-client.html

[13] Ethereum (2016). *Contracts* [Online] Available: http://www.ethdocs.org/en/latest/contracts-and-transactions/contracts.html

[14] T. Gerring (2016, Feb). *Cut and try: building a dream* [Online] Available: https://blog.ethereum.org/2016/02/09/cut-and-try-building-a-dream/

[15] Ethereum Stack Exchange (2016). *Ethereum Stack Exchange* [Online] Available: http://ethereum.stackexchange.com/

[16] Ethereum (2016). *Ethereum GitHub* [Online] Available: https://github.com/ethereum/

[17] EtherCasts (2016). *State of the Dapps* [Online] Available: http://dapps.ethercasts.com/

[18] Ethereum (2016). *Contracts and Transactions* [Online] Available: http://ethdocs.org/en/latest/contracts-and-transactions/index.html

[19] Ethstats (2016). *Ethereum Network Status* [Online] Available: https://ethstats.net/

[20] Ethereum Stack Exchange (2016). *How does Ethereum deal with blockchain scalability?* [Online] Available: http://ethereum.stackexchange.com/questions/133/how-does-ethereum-deal-with-blockchain-scalability

[21] Ethereum Reddit (2015). *REMINDER: the block gas limit is currently 5000, so it is not possible to send transactions. We will release an update in a couple days to cause miners to "vote" the gas limit upwards at which point the full blockchain functionality will be de-facto enabled.* [Online] Available: https://www.reddit.com/r/ethereum/comments/3f6f1r/reminder_the_block_gas_limit_is_currently_5000_so/

[22] Ethereum (2016). *Mining* [Online] Available: http://www.ethdocs.org/en/latest/mining.html

[23] G. Wood, "Ethereum: A secure decentralised generalised transactions ledger: Homestead draft", 2014. [Online] Available: http://gavwood.com/Paper.pdf Accessed: May 18, 2016.

[24] Ethereum Stack Exchange (2016). *What proof of work function does Ethereum use?* [Online] Available: http://ethereum.stackexchange.com/questions/14/what-proof-of-work-function-does-ethereum-use

[25] Vlad Zamfir (2015). *Introducing Casper "the Friendly Ghost"* [Online] Available: https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/

[26] Ethereum Reddit (2015). *WTF are uncles and why do they matter?* [Online] Available: https://www.reddit.com/r/ethereum/comments/3c9jbf/wtf_are_uncles_and_why_do_they_matter/

[27] Ethereum (2015). *What is Ether* [Online] Available: https://www.ethereum.org/ether

[28] Ethereum (2015). *Frequently Asked Questions* [Online] Available: http://www.ethdocs.org/en/latest/frequently-asked-questions/frequently-asked-questions.html

[29] IBM Open Blockchain (2016). *Open Blockchain Docs* [Online] Available: https://github.com/openblockchain/obc-docs

[30] IBM Open Blockchain (2016). *Open Blockchain Whitepaper* [Online] Available: https://github.com/openblockchain/obc-docs/blob/master/whitepaper.md

[31] IBM Open Blockchain (2016). *Canonical Use Cases* [Online] Available: https://github.com/openblockchain/obc-docs/blob/master/biz/usecases.md

[32] IBM Open Blockchain (2016). *Contributors to Open Blockchain* [Online] Available: https://github.com/openblockchain/obc-peer/graphs/contributors

[33] IBM Open Blockchain (2016). *Usage* [Online] Available: https://github.com/openblockchain/obc-docs/blob/master/FAQ/usage_FAQ.md

[34] IBM Open Blockchain (2016). *OBC Consensus Algorithm* [Online] Available: https://github.com/openblockchain/obc-docs/blob/master/FAQ/consensus_FAQ.md

[35] IBM Open Blockchain (2016). *Open Blockchain Protocol Specification* [Online] Available: https://github.com/openblockchain/obc-docs/blob/master/protocol-spec.md

[36] M. Castillo, CoinDesk (2016, March). *Hyperledger On Verge of Merging Blockchain IBM, Digital Asset Code* [Online] Available: http://www.coindesk.com/hyperledger-on-the-verge-of-merging-blockchain-code-from-ibm-digital-asset/

[37] IBM (2016). *IBM Blockchain* [Online] Available: http://www.ibm.com/blockchain/

[38] G. Maxwell, "Bringing New Elements to Bitcoin with Sidechains", transcript, *SF Bitcoin Devs Meetup*, 2015. [Online] Available: https://diyhpl.us/wiki/transcripts/gmaxwell-sidechains-elements/

[39] Elements Project (n.d.) *Confidential Transactions* [Online] Available: https://elementsproject.org/elements/confidential-transactions/

[40] Elements Project (n.d.) *Signature Covers Value* [Online] Available: https://elementsproject.org/elements/signature-covers-value/

[41] L. Parker, Brave New Coin (2015, June). *Blockstream Releases Sidechain Elements* [Online] Available: http://bravenewcoin.com/news/blockstream-releases-sidechain-elements/

[42] Elements Project (2015). *alpha-README.md* [Online] Available: https://github.com/ElementsProject/elements/blob/alpha/alpha-README.md

[43] Elements Project (2016). *Elements Project GitHub* [Online] Available: https://github.com/ElementsProject

[44] Elements Project (2016). *The Elements Project* [Online] Available: https://elementsproject.org/

[45] Blockstream (2015). *Fact Sheet* [Online] Available: https://blockstream.com/fact-sheet/

[46] Elements Project (2016). *Contributors to Elements Project* [Online] Available: https://github.com/ElementsProject/elements/graphs/contributors

[47] A. Wirdum, Bitcoin Magazine (2015, October). *Blockstream to Launch First, Instant-Settlement Sidechain for Bitcoin Exchanges* [Online] Available: https://bitcoinmagazine.com/articles/blockstream-to-launch-first-instant-settlement-sidechain-for-bitcoin-exchanges-1444755147

[48] Elements Project (n.d.). *Getting Started with the Alpha Sidechain* [Online] Available: https://elementsproject.org/sidechains/alpha/getting-started.html

[49] G. Prisco, Bitcoin Magazine (2015, June). *Blockstream Moves Ahead with Sidechain Elements, the First Implementation of Sidechains* [Online] Available: https://bitcoinmagazine.com/articles/blockstream-moves-ahead-sidechain-elements-first-implementation-sidechains-1433883105

[50] Intel Sawtooth Lake (2016). *Introduction* [Online] Available: http://intelledger.github.io/introduction.html

[51] Intel Sawtooth Lake (2016). *Distributed Ledger Web API* [Online] Available: http://intelledger.github.io/sawtooth_developers_guide/web_api/index.html

[52] Intel Sawtooth Lake (2016). *Sawtooth Lake Documentation* [Online] Available: http://intelledger.github.io/

[53] Intel Sawtooth Lake (2016). *Distributed Ledger* [Online] Available: https://github.com/IntelLedger

[54] Intel Sawtooth Lake (2016). *Architecture Overview* [Online] Available: http://intelledger.github.io/sawtooth_developers_guide/architecture_overview.html

[55] M. Hoekstra, Intel (2015). *Intel SGX for Dummies (Intel SGX Design Objectives)* [Online] Available: https://software.intel.com/en-us/blogs/2013/09/26/protecting-application-secrets-with-intel-sgx

[56] R. Chirgwin, The Register (2016, February). *Intel's SGX security extensions: Secure until you look at the detail* [Online] Available: http://www.theregister.co.uk/2016/02/01/sgx_secure_until_you_look_at_the_detail/

[57] V. Costan and S. Devadas, "Intel SGX Explained", Cryptology ePrint Archive, Rep. 2016/086, 2016. [Online] Available: http://eprint.iacr.org/2016/086.pdf

[58] S. Davenport and R. Ford, Virus Bulletin (2014). *SGX: the good, the bad and the downright ugly* [Online] Available: https://www.virusbulletin.com/virusbulletin/2014/01/sgx-good-bad-and-downright-ugly

[59] Eris (2016). *Documentation — eris:db Permissions* [Online] Available: https://docs.erisindustries.com/documentation/eris-db-permissions/

[60] Eris (2016). *Tutorials — The Eris Stack Explained* [Online] Available: https://docs.erisindustries.com/explainers/the-eris-stack/

[61] Eris (2016). *Eris Documentation Home* [Online] Available: https://docs.erisindustries.com/

[62] P. Byrne, Eris (2014, December). *Meet Eris. The distributed computing solution for industry.* [Blog] Available: https://blog.erisindustries.com/products/2014/12/17/eris-the-corporate-view/

[63] G. Prisco, Bitcoin Magazine (2016, February). *PwC Partners with Blockstream and Eris Industries to Create Blockchain Solution Portfolio* [Online] Available: https://bitcoinmagazine.com/articles/pwc-partners-with-blockstream-and-eris-industries-to-create-blockchain-solution-portfolio-1454359482

[64] J. O'Connell, Crypto Coins News (2016, March). *Eris and Ledger Form Partnership To Further Blockchain Technology* [Online] Available: https://www.cryptocoinsnews.com/eris-ledger-blockchain/

[65] PRWeb (2016, April). *BigchainDB and Eris Industries Partner to Build an Enterprise-Grade Decentralized Application Stack* [Online] Available: http://www.prweb.com/releases/bigchaindb/erisindustries/prweb13338588.htm

[66] Eris Industries (2016). *Eris Industries GitHub* [Online] Available: https://github.com/eris-ltd

[67] J. Kwon, *Tendermint: Consensus without Mining*. [Online] Available: http://tendermint.com/docs/tendermint.pdf

[68] Eris (2016). *Explainer — Blockchains* [Online] Available: https://docs.erisindustries.com/explainers/blockchains/

Appendix - COMS4507 Implementation Report

R. Julien, M. Macdonald, L. Liu-Thorrold

May 30, 2016

Abstract

Although the primary objective of this project was to perform a comparison across blockchain platforms available for use in general purpose applications, we wanted to explore how practical it was to utilise such technologies in everyday applications. This report details our implementation of a simple reputation system, managed on the backend by an in-memory blockchain on the Ethereum platform and demonstrated via a "Stack Overflow Style" web application.

1 Background and Related Works

At the time of writing, there is strong interest in utilising the technology underpinning the successful digital currency Bitcoin in applications where similar properties are desired (e.g. immutability, forgery resistance, decentralised management and anonymity). One type of application blockchain could potentially be suited for is a reputation system, as described in section 3.3 of the report.

Some literature exists in this field and include "Rep on the block: A next generation reputation system based on the blockchain" by R. Dennis and G. Owen - refer to [2] in the references. Their proposal for a reputation system is summarised in section 3.3 of the report. A patent has also been filed by Dr. Paul Ashley of Anonyome Labs Inc. titled "Decentralised Reputation Service for Synthetic Identities".

2 Design and Implementation

Our team has implemented a simple web application and contract, which can interact when the contract is deployed on the Ethereum blockchain platform. The web application is "Stack Overflow Style" where users are able to create an account, create posts, respond to comments, and vote on a post or comment. The vote is reflected through either an increase of decrease in the reputation of the author of that post or comment. Users are able to view the name of the account (identified by their public key) who upvoted or downvoted a post or comment.

3 Testing, Demonstration

The general development workflow for developing a distributed application is to test the distributed application and its contract on a private blockchain network,

or some other framework that allows the developer to test interaction with a full Ethereum client through simulation. For the purposes of demonstration, an in-memory blockchain will be used. This is provided by testrpc, a fast Ethereum RPC client for testing and developing distributed applications created using Ethereum [1].

Deployment on the real Ethereum network requires the syncing of over 9 million blocks, plus Ether to pay for gas which is required for maintaining the contract on the Ethereum blockchain network. The gas can be obtained from mining once the full blockchain has synchronised, or it can be purchased. There is an Ethereum network designed specifically for testing contracts that is available to the general public - however this still required the synchronisation of over 1 million blocks. Given that there are less users in the test network it is much smaller, however the time required to synchronise the 1 million blocks was still considerable, and mining was still required to obtain ether for the deployment.

Therefore, for the purposes of testing and demonstrating our application, testrpc was deemed sufficient.

4 Evaluation

Ethereum was selected for developing our application as it is one of the most capable and mature blockchain platforms available - as mentioned in section 6 of the report. It did not take a lot of time to get a simple application up and running, and appeared to be best suited for general purpose applications. This is in comparison to over half of the blockchain platforms studied which were aimed at developing enterprise applications. The remainder were either highly experimental, or not ready to use.

A few challenges we faced were caused by the youth of the project as well as its rapid development. For example, some tutorials written less than 3 months ago are already invalid due to changes in the API of Ethereum. This was a consistent theme throughout the project - it was difficult to find documentation that was up to date and that we could follow, some API methods have not yet been implemented, and there were errors in the code, introduced by developers, which is unfortunately to be expected in an open source community.

5 Future Works

Given that this was simply a proof of concept reputation application developed with the Ethereum platform, there are considerable improvements that can be made. Enforcing integrity in the application through the blockchain platform would be the most obvious start. It could eventually be deployed using cloud services to a private or semi-private Ethereum blockchain environment where it would be operated and maintained by the cloud service provider. These services are advertised as available by IBM, and Microsoft Azure [2],[3].

References

[1] ethereumjs, “ethereumjs/testrpc”, GitHub, 2016. [Online]. Available: https://github.com/ethereumjs/testrpc. Accessed: May 30, 2016.

[2] M. Gray,"Ethereum Blockchain as a service now on azure," 2015. [Online]. Available: https://azure.microsoft.com/en-us/blog/ethereum-blockchain-as-a-service-now-on-azure/. Accessed: May 30, 2016.

[3] "IBM Bluemix Docs,". [Online]. Available: https://console.ng.bluemix.net/docs/services/blockchain/index.html. Accessed: May 30, 2016.