SCTE·ISBE CABLE-TEC EXPO 2017
DENVER, CO
OCTOBER 17-20

2017 Fall Technical Forum
SCTE·ISBE • NCTA • CABLELABS

Comparing Blockchain Implementations

A Technical Paper prepared for SCTE/ISBE by

Zane Hintzman
Associate Engineer
Employee/CableLabs
11605 Destination Drive, Apt 5207
Broomfield, CO 80021
303-517-2664
z.hintzman@cablelabs.com

Table of Contents

List of Tables

Introduction

Blockchain technology is now very popular because it provides a new tool to solve problems in a way people could not before. When people think of blockchains, they most likely think of Bitcoin, the most well-known implementation of a blockchain. There are many other blockchain implementations as well. Some of them are still in development, while others are currently running. Different implementations will vary in many ways such as their purpose, ease of participation, how governance is handled, and much more. To determine which blockchain implementation should be leveraged for a given application, it is important to be familiar with the differences between each implementation.

Possibly the most important consideration for a blockchain implementation is its purpose. This seems obvious, but is truly overlooked by many developers. Most existing blockchains are specialized for cryptocurrencies, and many of these blockchains use Bitcoin's codebase. These blockchains often provide ways to enable usage in non-monetary applications by storing application data within transactions. For example, some solutions store information off the blockchain, then create a hash of that information and submit it in a blockchain transaction. The hash is stored on the blockchain and can be referenced by anyone to validate the same information. However, these methods are often limited in use, so cryptocurrency blockchains might not be a good fit for other applications. This makes it very important to understand the purpose of the application utilizing a blockchain to ensure that it aligns with the intended usage of the implementation.

The next aspect to consider is the ease of participating in the blockchain. For example, connecting to a blockchain may require running a full node that stores the entire history of transactions and blocks. Some blockchains have lightweight clients that allow access to the blockchain's network without downloading the whole transaction history. In addition, it is necessary to check whether the blockchain ecosystem is open to anybody, or only a closed group. If the codebase is open-source, then it will typically be an open network. Developers can create a *software fork* from open-source code to make a similar but different blockchain implementation. But doing so will bring maintenance under one's own responsibility.

The governance of a blockchain can also impact how it operates and what it can do. Thus, one should consider who controls access to the ecosystem, and who enforces decisions to make changes to the blockchain. Depending on the implementation, it may or may not be possible to see who is making or enforcing these decisions. Most blockchains require consensus from all participants in order to agree to a change on the blockchain. Someone who has less control in the blockchain's network may find it difficult to utilize the blockchain for their specific application.

Another important aspect is how well the blockchain performs. This means measuring how fast transactions are accepted by the blockchain's network, how much bandwidth it uses, how much blockchain data needs to be stored, and in what way the data must be stored. More specific metrics that one should measure are how quickly blocks are added to the blockchain, block and transaction sizes, and transaction rates. It is also important to note how transactions are bundled within blocks, and what limitations that may cause.

Some people are interested in blockchain technology because of the enhanced security features they offer. Different blockchains often use varying algorithms for key parts of their systems. For example, there are many algorithms that can be used to create the proof-of-work or proof-of-stake necessary to add blocks to the blockchain. Blockchains may also use different scripting languages to run smart contracts. Some blockchains might implement a managed public key infrastructure (PKI) for strong validation of users.

Furthermore, they may have widely differing consensus models for obtaining agreement on the state of the blockchain.

This paper aims to compare many blockchain implementations based on these criteria. Bitcoin will be investigated first because many people are familiar with it. Then other implementations will be described and compared to each other, with Bitcoin as a starting point for discussion.

Comparison of Blockchain Implementations

1. Bitcoin

Bitcoin is a digital asset and payment system where users can perform transactions without any intermediary system. The original source code of Bitcoin includes an implementation of a blockchain. The Bitcoin blockchain stores data on transactions, which indicate the amount of currency that moved between two or more accounts. However, there are ways to encode additional data into transactions. Coinbase transactions, which are transactions that generate new currency, can encode 100 bytes of arbitrary data. But this is limited in use because it can only be used when a node successfully adds a new block. A more accessible option is to create a transaction that uses an OP_RETURN output. This type of transaction can encode 40 bytes of arbitrary data. Still, the primary focus of Bitcoin is digital currency transactions.

The main currency in Bitcoin is simply called *bitcoin* (BTC or Ƀ or ₿). The smallest unit of bitcoin is called a *satoshi*, named after Satoshi Nakamoto, the pseudonym for the person or persons who designed Bitcoin. One satoshi equals 1/100,000,000th of a bitcoin. Bitcoins are associated with Bitcoin addresses, and the Bitcoin network runs various scripts to validate the individual funds for each address. The scripts include both locking and unlocking scripts. Locking scripts specify the conditions for spending an output, whereas unlocking scripts contain information to satisfy the conditions of some locking script. Both kinds are written in a scripting language known simply as *script*. It is a stack-based language that is purposefully limited in scope to prevent bugs such as infinite loops.

The Bitcoin protocol is fully based on open-source software which can be downloaded from GitHub (https://github.com/bitcoin/bitcoin). The most common way that users participate on the blockchain is by exchanging bitcoin with other users. To get bitcoin, a user generates a private key from which they generate a public key. Then from the public key, they create a Bitcoin address to which others can send bitcoin. There are many third-party sources where bitcoin can be obtained at the current bitcoin exchange rate. A high-level description of this process is presented in a companion paper by Steve Goeringer, a Principal Architect at CableLabs (1).

Another way to interact with Bitcoin is to run a node, which can be run by anyone who has sufficiently powerful hardware. Users can run a full node, which stores the entire Bitcoin blockchain with the history of all bitcoin transactions. But running a full node requires significant hardware. It is recommended to have at least 145 GB of disk space and 2 GB of memory. Over time, more storage and memory will be required. A full node may also use a lot of CPU when it initially synchronizes with the Bitcoin network. An alternative is to run a lightweight client, also known as a simplified payment verification (SPV) node. This stores a user's wallet, but relies on third-party servers to view the blockchain and perform transactions. There are also web clients that allow you to see Bitcoin data on a web browser, but rely on third-party servers to interact with Bitcoin. Furthermore, the Bitcoin network has many application programming interface (API) calls that read from or write to the blockchain.

Bitcoin has two distinct blockchains: the main blockchain and the testnet blockchain. Both use the same underlying protocols, but only the main blockchain is considered a valid means of payment for real-world services. The purpose of the testnet is to allow people to test Bitcoin applications without potentially harming the main blockchain. The testnet is a good place to start testing blockchain applications to see if they can run efficiently with the Bitcoin protocol. Furthermore, Bitcoin also has a regression test mode, allowing users to create a private blockchain where they control when new blocks are created. This provides another environment for users to test Bitcoin blockchain applications. Refer to Bitcoin's website (https://bitcoin.org) for more information on how to connect to Bitcoin blockchains.

The Bitcoin blockchain is not owned by any single entity. All nodes in the blockchain's network have an impact on how the blockchain operates. Every transaction is transmitted to every full node, and all full nodes coordinate to obtain the same blockchain. The protocol can be updated across the network if the majority of full nodes agree to implement a particular change. All users, whether they run a full node or a lightweight node, are anonymous to the network. A user's Bitcoin address is used to distinguish different accounts in the network, and a user's wallet can contain more than one address. But there is no requirement to associate personal information with one's Bitcoin addresses. Therefore, it is not directly possible to find out who (meaning real people, not Bitcoin addresses) is participating in the Bitcoin network. Every node contributes to the network and aims to achieve consensus with other nodes.

Consensus is achieved through four general processes. First, each full node independently verifies each transaction based on several criteria. Second, full nodes work independently to aggregate transactions into blocks so they can earn a bitcoin reward. Third, all full nodes verify new blocks as they come in. Finally, full nodes independently select a blockchain with the most cumulative computation demonstrated through proof-of-work. The last step means nodes will choose the blockchain with the most blocks that have a valid proof-of-work.

New bitcoins are generated through a process called *mining*, and is performed by nodes called *miners*. First, a miner consolidates one or more transactions into a block. Next, they calculate a proof-of-work for that block so it can be approved by the network. This is done by creating a hash of the block. As long as the hash is not numerically less than the current difficulty target, the miner changes the *nonce* field in the block so the block's hash also changes. Once the hash is less than the target, the miner transmits the block across the Bitcoin network. Lastly, if the block is accepted by the entire network, the miner receives a reward in the form of newly generated bitcoin. When this happens, miners can no longer submit blocks at the same height as the accepted block. Thus, miners will want to complete these steps faster than other miners to get bitcoin rewards and avoid wasted effort. The most compute-intensive step is calculating the proof-of-work for the block. Miners that can compute a proof-of-work faster than other miners will have a better chance of getting bitcoin rewards.

The protocol is specifically designed so that it always takes around 10 minutes to generate a proof-of-work. This determines the minimum time to create a new block, sometimes called the block-release timing. After a specific number of blocks, the network will change the difficulty of creating a proof-of-work to ensure the process still takes about 10 minutes. The block-release timing is set to 10 minutes in order to decrease the probability of a *fork*, where there are multiple blockchains competing to be considered the main blockchain. The protocol is also limited in that the maximum size a block can have is 1 MB, and the minimum size of a transaction is around 200 bytes. Based on these constraints, the maximum theoretical transaction rate is 7 transactions per second. In contrast, Amazon can handle 1 million transactions per second on most days, and can process 600 transactions per second on days with

lots of internet traffic. Thus, Bitcoin may limit the number of transactions per second for large scale companies like Google and Amazon.

Bitcoin uses a few different algorithms for various aspects of the protocol. To generate a public key from a private key, it uses the Elliptic Curve Digital Signature Algorithm (ECDSA). According to the book *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*, "Bitcoin uses a specific elliptic curve and set of mathematical constraints, as defined in a standard called secp256k1, established by the National Institute of Standards and Technology (NIST)" (2). To create a Bitcoin address, the algorithms SHA256 and RIPEMD160 are used sequentially to hash a public key into the Bitcoin address. SHA256 is also used to calculate a block's hash, and thus the proof-of-work for the block.

As the most well-known implementation of a blockchain, Bitcoin is a good starting point for comparing other implementations. But just because Bitcoin is the most famous implementation does not mean it is the best. There are many other cryptocurrency blockchains that make some improvements to the original protocol. In addition, blockchains that are not focused on cryptocurrency are more applicable to other use cases.

1.1. Summary of Bitcoin

Table 1 – Summary of Bitcoin blockchain

	Bitcoin
Purpose	Cryptocurrency
What kind of data can be stored?	Cryptocurrency transactions, plus some additional data in coinbase or OP_RETURN transactions
Scripting Languages	Script
Is the ecosystem open?	Yes
How can one participate?	Download the source code from GitHub, and follow their instructions. Obtain currency from online trading service.
Native Currency	bitcoin (BTC or Ƀ or ₿)
Who are the registration authorities?	N/A
Is decision making transparent?	Yes
Does it use a managed PKI?	No
Who manages PKI?	N/A
Block-release Timing	10 minutes
Transaction Size	200 bytes minimum, 250 bytes avg.
Transaction Rate	3 tx/sec. avg., 7 tx/sec. theoretical maximum
Consensus Model	Nodes verify blocks and transactions, and select blockchain with the most blocks.
Mining	Proof-of-work

2. Alt Coins

The invention of Bitcoin inspired the creation of alternative decentralized currencies known as *alt coins*. One example is Litecoin, an alt coin derived from Bitcoin's source code. This makes Litecoin a software fork of Bitcoin, as is the case with most but not all alt coins. Litecoin runs on a separate blockchain that contains the entire history of all Litecoin transactions. As a software fork, Litecoin has many similarities with Bitcoin. For example, as with Bitcoin, Litecoin is an open-source protocol that specializes in cryptocurrency. Litecoin uses the same scripting language as Bitcoin, so transactions can store additional data using an OP_RETURN output script. In addition, the Litecoin blockchain can be accessed in the same way as Bitcoin: by trading for Litecoin's currency, *litecoin* (LTC), or by running a node.

However, Litecoin also has many differences from the original Bitcoin protocol. First, the time to calculate a proof-of-work for a block on the Litecoin blockchain is around 2.5 minutes, compared to 10 minutes in the Bitcoin network. Consequently, Litecoin can theoretically confirm transactions four times as fast as Bitcoin, making the theoretical maximum transaction rate 28 transactions per second. Next, Litecoin uses a different proof-of-work algorithm. Bitcoin uses SHA-256, whereas Litecoin uses *scrypt* (not to be confused with *script*, the scripting language of Bitcoin). It is said that scrypt is quicker and much simpler than SHA-256, but that it could be less secure. Furthermore, there are not as many lightweight client nodes for Litecoin as there are for Bitcoin. Litecoin has the Electrum Lightweight Litecoin Client (https://electrum-ltc.org), but it is still in the beta phase and few servers are running this client. This makes it slightly more difficult to join the Litecoin network. However, there are plenty of online services for storing litecoin, so it is not necessary to run a full node to obtain and use litecoin.

Yet another software fork of the Bitcoin protocol is Dogecoin. Dogecoin is also an open-source protocol specialized for cryptocurrency and runs on its own blockchain. As with Bitcoin, it is possible to join the Dogecoin network with a full node, a wallet client, or a lightweight client. The most significant difference between the Dogecoin and Bitcoin protocols is that it is much easier to generate new currency in Dogecoin. Whereas Bitcoin will stop creating new bitcoins once 21 million are created, the supply of *Dogecoin* (DOGE), the protocol's currency, is uncapped. Another difference to Bitcoin is that it takes much less time to create a proof-of-work for a block on the Dogecoin blockchain, taking only 60 seconds. This means that the Dogecoin network can theoretically confirm up to 70 transactions per second. Furthermore, Dogecoin uses scrypt for its proof-of-work algorithm, similar to Litecoin.

While the Litecoin and Dogecoin networks can confirm transactions faster than Bitcoin, they are still limited because they are specialized for digital currencies. Their transactions use the same fields as Bitcoin to store metadata. Saving only a small amount of metadata in transactions might not provide enough storage to leverage for non-cryptocurrency applications.

2.1. Summary of Alt Coins

Table 2 – Summary of alt coin blockchains

	Litecoin	Dogecoin
Purpose	Cryptocurrency	Cryptocurrency
What kind of data can be stored?	Cryptocurrency transactions, plus some additional data in coinbase or OP_RETURN transactions	Cryptocurrency transactions, plus some additional data in coinbase or OP_RETURN transactions
Scripting Languages	Script	Script
Is the ecosystem open?	Yes	Yes
How can one participate?	Download the source code from GitHub, and follow their instructions. Obtain currency from online trading service.	Download the source code from GitHub, and follow their instructions. Obtain currency from online trading service.
Native Currency	litecoin (LTC)	Dogecoin (DOGE)
Who are the registration authorities?	N/A	N/A
Is decision making transparent?	Yes	Yes
Does it use a managed PKI?	No	No
Who manages PKI?	N/A	N/A
Block-release Timing	2.5 minutes	60 sec.
Transaction Size	(unknown)	(unknown)
Transaction Rate	28 transactions/sec. theoretical maximum	70 transactions/sec. theoretical maximum
Consensus Model	Nodes verify blocks and transactions, and select blockchain with the most blocks.	Nodes verify blocks and transactions, and select blockchain with the most blocks.
Mining	Proof-of-work	Proof-of-work

3. Ethereum

Not all blockchain implementations are specialized for cryptocurrency. The Ethereum protocol, for example, is a decentralized blockchain platform for running smart contracts. Smart contracts are small executable programs that are run only if certain conditions are met. Not only can Ethereum create new cryptocurrencies, but also it can create digital tokens representing something like a physical asset, a virtual share, or a proof of membership. It is also possible to run a virtual organization through Ethereum by controlling processes in the organization with smart contracts. While the Bitcoin blockchain contains a list of transactions, Ethereum tracks the state of different accounts. There are two types of accounts in the Ethereum network. The first is an *externally owned account* (EOA), which is controlled by a private key. The other kind is a *contract account*, which are snippets of code usually written in some high-level coding language. A contract account is controlled by its contract code and can only be activated by an EOA.

Ethereum runs on its own blockchain and has its own cryptocurrency, called “ether,” to use as payment for submitting transactions. Ether is used to pay for “gas,” which is the internal pricing for running a

transaction or contract in Ethereum. The amount of gas needed to commit a contract increases with the number of computational steps needed to run the contract. An important aspect of Ethereum is the Ethereum Virtual Machine (EVM), the runtime environment for smart contracts. All nodes in the network run the EVM and communicate with each other to execute the same instructions and maintain consensus. The primary language for writing contracts is Solidity, an object-oriented language designed to compile code for the EVM. Contracts can also be written in Serpent or Low-level Lisp-like Language (LLL), but these are less supported than Solidity.

As with Bitcoin, the Ethereum protocol is open-sourced on GitHub (https://github.com/ethereum), and the Ethereum network is open to anyone who can run a full node or use a wallet application. The Light Ethereum Subprotocol (LES) is currently in development to allow lightweight clients to run in Ethereum. There is also a public testnet blockchain for testing Ethereum applications, and the protocol allows you to create a private testnet blockchain for internal testing.

Blocks are added to Ethereum's blockchain similarly to Bitcoin. Ethereum uses its own algorithm called *Ethash* to create a proof-of-work for blocks. Transaction latency for Ethereum is only 12 seconds per block, which is significantly faster than the Bitcoin network. Furthermore, the maximum size of a block or transaction is limited only by the amount of gas in circulation. Since the block gas limit is about 3 million gas, the maximum block or transaction size is around 89 kB. The amount of gas necessary to perform a single transaction can vary greatly, but it cannot exceed 3 million gas.

While Ethereum does have its own cryptocurrency, the main purpose of the network is to run smart contracts. This makes it applicable to a larger set of use cases than Bitcoin. However, recent events have created some doubts about the stability of Ethereum. Refer to section 3.2 for details on these events. More information on the Ethereum protocol is available on the main webpage (https://www.ethereum.org) and the documentation webpage (http://ethdocs.org).

3.1. Summary of Ethereum

Table 3 – Summary of Ethereum blockchain

	Ethereum
Purpose	Run smart contracts
What kind of data can be stored?	Cryptocurrency, digital assets, smart contracts
Scripting Languages	Solidity, Serpent, LLL
Is the ecosystem open?	Yes
How can one participate?	Download the source code from GitHub, and follow their instructions. Obtain currency from online trading service.
Native Currency	ether (ETH or ETC)
Who are the registration authorities?	N/A
Is decision making transparent?	Yes
Does it use a managed PKI?	No
Who manages PKI?	N/A
Block-release Timing	12 sec.
Transaction Size	Theoretically no max (actual max: 89 kB)
Transaction Rate	Theoretically no maximum
Consensus Model	Similar to Bitcoin, but uses Ethereum Virtual Machine
Mining	Proof-of-work using Ethash algorithm

3.2. Ethereum Network Hack

As usage of a blockchain network goes up, the chance of it becoming the target of hackers increases. An example is the recent hack on the Ethereum blockchain. On June 17th, 2016, a hacker stole over $50 million dollars' worth of ether (3.6 million ether) from the Decentralized Autonomous Organization (D.A.O.), an experimental virtual currency project that had raised $160 million in the form of ether.

It appears that the hack was not caused by any vulnerability in the Ethereum codebase. Rather, it was caused by a software vulnerability in the D.A.O.'s code. On June 9th, the vulnerability that was exploited on the 17th was noted by Petter Vessenes in a blog post. Furthermore, on May 27th, a group of computer scientists released a paper that explained many vulnerabilities in the D.A.O. Emin Gün Sirer, a co-author of this paper, noted that it's easy to make mistakes when using Solidity to code smart contracts. In short, the exploited vulnerabilities were public information before the hack even occurred.

A few hours after the hack, the price of ether dropped from $21.50 per unit to $15 per unit. A brief time later, Vitalik Buterin, the founder of Ethereum, proposed a *soft fork* in the Ethereum network to prevent the attacker from using the stolen funds for another 27 days. This was followed by a *hard fork* on July 20th to allow users to recover their ether. Buterin assumed that the majority of nodes would conform to the rules of the new hard fork, but many nodes remained separate from the hard fork. As a result, there are now two competing Ethereum blockchains: the first is simply called Ethereum (ETH), and the second is known as Ethereum Classic (ETC). The funds stolen were returned to the D.A.O. in the ETH network, but were not returned in the ETC network. Some people believe that having two networks will allow for "replay attacks." This means that when a user tries to send a transaction to one of these networks, they might accidentally send the same transaction to the other network as well.

This event demonstrates the importance of security in blockchains. When evaluating a blockchain, one should make sure the network is secure, and that there are no obvious flaws. However, the security of user clients, wallets, exchanges, and scripting languages must also be carefully engineered and monitored. In the case of this hack, the fault was not even in the codebase of Ethereum itself. Yet the network was still affected because many people implemented the hard fork to protect the investors of the D.A.O. As a network becomes popular and the value of transactions increase, it will become more and more attractive to cyber thieves and activists.

It is also important to look at the current state of the blockchain, regardless of whether it has been hacked. The hack on the Ethereum network resulted in two competing versions of the Ethereum blockchain, making it much harder and more confusing to participate. It is currently unclear which version will be more widely adopted. Thus, it may be beneficial to avoid blockchains like Ethereum that have been damaged by attacks.

4. MultiChain

An application running on the Bitcoin or Ethereum blockchains must conform to their respective protocols in order to participate. However, it is possible to obtain more implementation freedom with MultiChain. MultiChain is a platform for creating private blockchains. Users define the parameters of the blockchain they create, and they can record multiple kinds of assets on their blockchain. The platform is available to download and install on Linux or Windows machines, and the source code is available on GitHub (https://github.com/MultiChain/multichain). MultiChain extends the Bitcoin APIs, and has a

similar protocol and transaction format. A node set up through MultiChain can also act as a node on the Bitcoin network or the Bitcoin testnet network.

For blockchains created using MultiChain, the protocol allows creators to determine what permissions a new participant will have without receiving them from an administrator. Before the blockchain is initialized, the creator determines the initial set of administrators, as well as whether anyone can connect to the network without restriction. Administrators can also dynamically control permissions to the blockchain for specific users while the blockchain is active. Such permissions include the ability to send, receive, or create assets, and the ability to create blocks. Decisions to alter permissions are made via consensus among administrators. The proportion of permitted administrators who must agree to modify a user's privilege is set before the blockchain starts running. This value can be fine-tuned for every kind of permission on the network.

Creators also control how fast the network moves and the size of the data in the blockchain. Again, these parameters are specified before creating the blockchain. The creator can specify the target for the average time to add a block, the maximum block and transaction sizes, and the maximum size of an OP_RETURN metadata output. They can also change the mining difficulty, how frequently the difficulty is updated, and even whether a proof-of-work is required to add new blocks. Regardless of these settings, the blockchain will use a randomized round-robin system to add blocks. The protocol relies on another parameter called *mining diversity*. This determines the minimum proportion of miners required to participate in round-robin mining in order to render a valid blockchain. When a miner adds a new block to the blockchain, it must wait for this minimum proportion of other miners to add blocks before it can add anymore blocks. This may significantly increase the Byzantine fault tolerance (BFT) of MultiChain blockchains in comparison to some other blockchains.

MultiChain offers a lot of flexibility in designing new blockchains, but it has some downsides. A significant limitation of MultiChain is that it does not implement smart contracts, nor does the company have immediate plans to implement smart contracts. Furthermore, although the source code for MultiChain was recently made public, the project is in beta at the time of this writing. The main Multichain webpage (http://www.multichain.com) contains materials with more details on the protocol.

4.1. Summary of MultiChain

Table 4 – Summary of MultiChain blockchains

	MultiChain
Purpose	Provide a platform for creating your own blockchain.
What kind of data can be stored?	Any digital asset you want to store.
Scripting Languages	N/A
Is the ecosystem open?	Configurable
How can one participate?	Install MultiChain app, and follow online instructions to make a blockchain.
Native Currency	N/A
Who are the registration authorities?	Configurable
Is decision making transparent?	Configurable
Does it use a managed PKI?	No
Who manages PKI?	N/A
Block-release Timing	Configurable

	MultiChain
Transaction Size	Maximum size configurable
Transaction Rate	Configurable
Consensus Model	Fixed ratio of admins approves privilege changes. Longest valid blockchain adopted as global consensus.
Mining	Round-robin system; proof-of-work requirement is configurable

5. Hyperledger

Many companies have come together to form Hyperledger, a Linux Foundation project whose goal is to advance blockchain technology to benefit a variety of business use cases. There are many projects associated with Hyperledger. Subsequent sections of this document will describe three of the Hyperledger frameworks: Hyperledger Fabric, Hyperledger Sawtooth, and Hyperledger Iroha. These projects provide users with the tools to deploy their own blockchains. A summary of all projects can be found at https://www.hyperledger.org/projects.

5.1. Hyperledger Fabric

Hyperledger Fabric is a blockchain implementation designed by IBM for industry use cases. A blockchain deployed using Hyperleger Fabric stores data in the form of *chaincode*, a programmatic code on the network that functions similar to smart contracts on other blockchains. The network currently supports Golang as the language for chaincode, and progress is being made to enable Java as well. By default, the network does not include its own native cryptocurrency. However, users can implement a cryptocurrency through chaincode.

The source code for Hyperledger Fabric is available on GitHub (https://github.com/hyperledger/fabric), and includes instructions on how to create a new blockchain using this implementation. A blockchain implemented using Hyperledger Fabric is a permissioned network. Thus, new participants must register with a proof of identity to the network membership services. Transactions sent by each user include derived certificates that cannot be linked to the sender. Each transaction has its content encrypted so it cannot be viewed by unintended participants. The initial registration authorities are set in a configuration file before the blockchain begins running. This file also determines which initial users can assign additional registration authorities while the network is running. The registration authorities are a part of the blockchain's membership services.

Membership services also include roles that implement a PKI. These roles include an enrollment certificate authority (ECA), a transaction certificate authority (TCA), and a transport layer security certificate authority (TLS-CA). An ECA issues enrollment certificates (ECerts) to network participants, a TCA issues transaction certificates (TCerts) to users with an ECert, and a TLS-CA issues TLS certificates to secure communication channels. Certificate authorities must be initialized before the network starts running so that new nodes will connect to the CA.

The performance of Hyperledger Fabric blockchains has not been extensively tested. However, it is expected that they can process transactions at a rate greater than 10k transactions per second using a BFT consensus model. Hyperledger Fabric has a pluggable consensus framework, which provides multiple options for the consensus algorithm used by the blockchain. Currently, the source code provides implementations for two different algorithms. The first implementation is for the Practical Byzantine

Fault Tolerance (PBFT) consensus protocol. The second is a "dummy" consensus protocol that doesn't perform consensus but still processes all consensus messages. The latter is simply for development and test purposes, and to provide an example of how to create a consensus model plugin.

Hyperledger Fabric has many benefits in theory, but it is still in development. Still, this establishes that people are working on other ways to implement blockchains. The success of this project would provide users another way to implement a blockchain and give them control over parameters in the blockchain's network. The documentation page (http://hyperledger-fabric.readthedocs.io) contains more details on the project.

5.2. Summary of Hyperledger Fabric

Table 5 – Summary of Hyperledger Fabric blockchains

	Hyperledger Fabric
Purpose	Enable the creation of blockchains for industry use cases.
What kind of data can be stored?	Chaincode (i.e. smart contracts)
Scripting Languages	Go (golang), Java (in progress)
Is the ecosystem open?	No
How can one participate?	Create a blockchain: Download source and follow instructions. Join existing network: Register with a proof of identity to the network membership services.
Native Currency	N/A
Who are the registration authorities?	Defined before blockchain is initialized, and more can be assigned while running.
Is decision making transparent?	(unknown)
Does it use a managed PKI?	Yes
Who manages PKI?	One or more entities in membership services.
Block-release Timing	(unknown)
Transaction Size	(unknown)
Transaction Rate	> 10k tx/sec.
Consensus Model	Pluggable consensus framework; 2 plugins provided: PBFT, and "dummy" plugin
Mining	N/A

5.3. Hyperledger Sawtooth

Hyperledger Sawtooth is a blockchain implementation published by Intel. The goal of the project is to provide companies a means to deploy their own blockchains. The type of data stored in transactions is determined by what the project calls a *transaction family*. Users can create their own transaction families to store any data they want to save in a blockchain deployed with Hyperledger Sawtooth. The source code includes three implementations of transaction families. One of these transaction families, called "MarketPlace," enables buying, selling, and trading digital assets.

The protocol supports both permissioned and permissionless implementations. Intel has released the source code on GitHub (https://github.com/hyperledger/sawtooth-core), and provides instructions on how to deploy a blockchain using Hyperledger Sawtooth. Transactions are transparent by default, but the

network can optionally use an administration key for sending certain messages. The amount of time needed to create a block of transactions is also configurable to any length of time.

The current source code includes implementations for two different consensus protocols. The first is called Proof of Elapsed Time (PoET). It is a lottery protocol that builds on trusted execution environments (TEEs) provided by Intel's Software Guard Extensions (SGXs). The protocol is based on the consensus algorithm used by Bitcoin. The other protocol is called "Quorum Voting," and is an adaption of the consensus protocols used by Ripple and Stellar. The current release of Hyperledger Sawtooth includes software that simulates the PoET algorithm, but this implementation of PoET is not secure because it runs outside of Intel's SGXs.

At this time, Intel warns against using Hyperledger Sawtooth for any security sensitive applications, as the project is still in the experimental phase. It will probably be a while before Hyperledger Sawtooth can be used to create secure blockchain applications. Hopefully, this project will provide another means to create blockchains. Documentation for this project is available at http://intelledger.github.io/.

5.4. Summary of Hyperledger Sawtooth

Table 6 – Summary of Hyperledger Sawtooth blockchains

	Hyperledger Sawtooth
Purpose	Enable companies to deploy their own blockchains.
What kind of data can be stored?	Anything that can be defined by a "transaction family".
Scripting Languages	Python
Is the ecosystem open?	Configurable
How can one participate?	Download the source code from GitHub and follow their instructions.
Native Currency	Initially provides the MarketPlace transaction family, which can track assets.
Who are the registration authorities?	None, unless optional administration key is used.
Is decision making transparent?	Yes; transaction transparency is default
Does it use a managed PKI?	No
Who manages PKI?	N/A
Block-release Timing	Configurable
Transaction Size	(unknown)
Transaction Rate	(unknown)
Consensus Model	Provides two: Proof of Elapsed Time (PoET), and Quorum Voting
Mining	N/A

5.5. Hyperledger Iroha

Hyperledger Iroha is a distributed ledger project designed to be easy to integrate into infrastructural projects. The main goal of this project is to provide C++, mobile, and web development environments to Hyperledger contributors. Few details have been provided on how this implementation works, but there is some information within the wiki pages of the GitHub repository for Hyperledger Iroha (https://github.com/hyperledger/iroha).

A blockchain deployed with Hyperleger Iroha can store two types of data: objects and functions. Stored functions are known as *chaincode*, and they can be set by users who have sufficient permissions. *Sumeragi* is the name of the Byzantine fault tolerant distributed consensus algorithm used by Hyperledger Iroha. Most of the algorithm is based on the B-Chain consensus algorithm. In sumeragi, consensus is performed on individual transactions and the global state resulting from all transactions. When a transaction is submitted to the blockchain, $2f+1$ signatures are needed to confirm a transaction, where f is the number of Byzantine faulty nodes the blockchain's network can handle. The order of nodes which validate a transaction is determined by a server reputation system called *hijiri*. The hijiri reputation system calculates the reliability of each server based on the time they were registered with membership services, the number of successful transactions they've processed, and any failures that are detected on the server. Hijiri performs the following tests on each server:

- A data throughput test
- A version test
- A computational test
- A data consistency test

The developers of Hyperledger Iroha claim that the platform will provide transaction finality within two seconds.

In short, Hyperledger Iroha will provide developers with more options for environments through which they can contribute to Hyperledger.

5.6. Summary of Hyperledger Iroha

Table 7 – Summary of Hyperledger Iroha blockchains

	Hyperledger Iroha
Purpose	Provide tools that integrate easily into existing environments.
What kind of data can be stored?	(unknown)
Scripting Languages	(unknown)
Is the ecosystem open?	(unknown)
How can one participate?	(unknown)
Native Currency	(unknown)
Who are the registration authorities?	(unknown)
Is decision making transparent?	(unknown)
Does it use a managed PKI?	(unknown)
Who manages PKI?	(unknown)
Block-release Timing	2 seconds
Transaction Size	(unknown)
Transaction Rate	(unknown)
Consensus Model	Sumeragi
Mining	(unknown)

6. Steem

Steemit is a blockchain-based social media platform where users earn rewards for posting content that the community considers meaningful. The network stores posts, votes, comments, profiles and follows on the *Steem* blockchain. Steem is built on Graphene, a software platform for deploying application-specific

blockchains. Source code for the Steem network is available on GitHub (https://github.com/steemit/steem), and there are instructions on how to download the blockchain locally. Anyone who simply wants to make posts on Steemit can create an account for free that is linked to their Facebook or Reddit account. Posts can be viewed at https://steemit.com, and the blockchain can be viewed at https://steemd.com. The latter site allows users to view all votes on a specific post or comment, and who made what votes. Users can interact with Steem in more ways by obtaining one or more of the cryptocurrencies underlying the network.

Steem has three different cryptocurrencies: Steem (STEEM), Steem Power (SP), and Steem Dollars (SMD). STEEM is the main unit of currency in the Steem network. The main purpose of STEEM is to easily convert into either SP or SMD. STEEM allows users to maintain liquidity for short periods of time. Steem Power is used to vote for or against content posted in the Steem network. Steem Dollars are meant to be pegged to the US dollar, and can be used to pay for goods or services. The Steem website also allows users to buy STEEM or SP using bitcoins. Users can convert between STEEM and SMD at the current exchange rate on a cryptocurrency exchange website, including Steem's own site at https://steemit.com/market.

When a post is made, voting periods are set by the blockchain, after which a payout is made to the user who created the post. The payouts are 50% SMD and 50% SP. From the wallet page on the Steem website, users can convert STEEM into SP, which is known as “powering up”, or convert SP into STEEM, which is referred to as “powering down”. Powering up can happen instantaneously, but when powering down, SP is converted to STEEM over two years via 104 equal weekly payments.

Steem uses a Delegated Proof-of-Stake (DPOS) consensus model. In this model, blocks are produced in separate rounds, each of which has 21 witnesses selected to create and sign blocks. Nineteen of these witnesses are chosen by approval voting, an additional one is selected by a computational proof-of-work, and the last is timeshared by every witness that wasn't in the top 19. Each round, a miner is taken from a queue of miners and added to the active set of witnesses. The miner can then earn a reward if they produce a block while they are a scheduled witness. Like most blockchain networks, the miner must create a proof-of-work to create a block of transactions. The calculation uses the SHA256 hashing algorithm twice. Below is an algorithm showing how the proof-of-work is calculated:

Let H = Head Block ID

Let H2 = SHA256(H+NONCE)

Let PRI = Producer Private Key

Let PUB = Producer Public Key

Let S = SIGN(PRI, SHA256(H))

Let K = RECOVER_PUBLIC_KEY(H2, S)

Let POW = SHA256(K)

POW is the value of the proof-of-work necessary to mine a block. Once a miner creates this value, they earn a reward in the form of SP.

The Steem network out performs Bitcoin in terms of transaction rate and transaction size. Steem is designed to create a block every three seconds, as opposed to 10 minutes for Bitcoin. Also, because Steem is built on Graphene, the same technology used by BitShares, Steem can process over 10,000 transactions per second, which is much faster than the Bitcoin network. Furthermore, the average size of a transaction is 250 bytes in Bitcoin, but is only 100 bytes for Steem.

Steem provides a unique implementation, supporting a use case for a targeted audience. While Steem is limited in the type of application data it can store, it provides more options than Bitcoin. The Steemit application is still in beta, yet it is already in wide use. This network shows great promise for usability with a variety of applications, but there could be some doubts for reasons that will be explained next. Refer to the whitepaper, *Steem: An incentivized, blockchain-based social media platform,* for more information on the Steem protocol. More details on the Steem project can be found at https://steem.io/.

6.1. Summary of Steem

Table 8 – Summary of Steem blockchain

	Steem
Purpose	Social media platform that rewards users for meaningful posts.
What kind of data can be stored?	Posts (text and pictures), votes, comments, profiles, and follows
Scripting Languages	(unknown)
Is the ecosystem open?	Yes
How can one participate?	Steemit: Make an account online and obtain assets. Blockchain: Download source code and follow their instructions.
Native Currency	Steem (STEEM), Steem Power (SP), and Steem Dollars (SMD)
Who are the registration authorities?	N/A
Is decision making transparent?	You know who upvotes what, but forking process isn't clear.
Does it use a managed PKI?	No
Who manages PKI?	N/A
Block-release Timing	3 seconds
Transaction Size	100 bytes avg.
Transaction Rate	10k transactions/sec.
Consensus Model	Delegated Proof-of-Stake
Mining	Proof-of-work

6.2. Steemit Network Hack

Users of the Steem network were recently hacked. On July 14th, 2016, around 260 accounts were compromised, with nearly $85,000 worth of Steem Dollars and Steem stolen.

The day of the hack, some users noticed that Steem funds from their accounts on Bittrex, a US-based cryptocurrency exchange, were mysteriously transferred to an unknown Bittrex account. According to Steem CEO Ned Scott, "…the Steem blockchain was never hacked. Likewise, our servers were never hacked. Instead, the hacker exploited browser-side vulnerabilities…" (3). After the hack, Steem programmers worked on preventative measures against this kind of attack. But immediately after this issue was resolved, Steem servers were hit by a DDoS attack, at which time the Steemit site was taken down to mitigate the damage.

On July 17th, the Steem producers released a post describing a new approach to securing user accounts through multi-factor authentication. The post was likely in response to the attack on Steem's network. Then on July 19th, the Steem producers described a process to recover accounts for users whose accounts were affected by the July 14th hack. The process required users to remember a password that was valid in their account within the last 30 days.

This hack demonstrates that relatively new networks can be a target for attacks. Steem is still considered to be in beta at the time of writing this report, and it has already been targeted by two related attacks. In the case of the first attack, the flaw was not in the Steem network itself. Rather, it was reportedly in the browser interface with Bittrex. This is similar to the Ethereum hack, where a flaw in an organization utilizing Ethereum was compromised. Thus, it goes to show that whether the blockchain network itself is secure is meaningless if the software that uses the network is insecure. The possibility of a system exploit should be taken into initial consideration when deciding on a blockchain implementation.

7. Sidechains – Elements Project

On October 22nd, 2014, a group of people released a paper titled *Enabling Blockchain Innovations with Pegged Sidechains*. The paper proposes an innovative technology called *pegged sidechains*, blockchains that are linked to another blockchain. In the paper, they define a *sidechain* as "a blockchain that validates data from other blockchains," then they define a pegged sidechain as "a sidechain whose assets can be imported from and returned to other chains; that is, a sidechain that supports two-way pegged assets" (4). Later that year, the same group of people formed the company Blockstream, whose goal is to enable the creation of sidechains. Then on June 6th, 2015, Blockstream announced the release of Sidechain Elements, a project that includes working code and a testing environment for creating pegged sidechains. This was later renamed as the Elements Project.

There are two ways to work with the Elements Project: one can either join the Alpha experimental sidechain, or they can create their own sidechain. Both can be done with the source code available on GitHub (https://github.com/ElementsProject/elements). The Elements Alpha sidechain focuses on cryptocurrency transactions, and uses a scripting language like that of Bitcoin but with several improvements. The Alpha sidechain is pegged to the Bitcoin testnet, as is any sidechain created using Elements. Currently, the only assets that these sidechains will track are its own native currency called *hostcoin*. The ability to pay using other assets is being developed.

The performance of either the Alpha sidechain or any pegged sidechains that users can create has not yet been tested. However, when creating one's own sidechain, by default the time to create a block is 60 seconds. This time can be modified in the appropriate configuration file.

As mentioned on the Blockstream website, the Elements Project is intended for research and development. Thus, this project should not be used with real-world assets, and is not yet fully secure. While this project is not yet complete, it demonstrates how sidechains could eventually become another way to implement a blockchain. The Elements Project is described in more detail on its main page (https://www.elementsproject.org). The specifics of the protocol are described in the whitepaper, *Enabling Blockchain Innovations with Pegged Sidechains*.

7.1. Summary of Elements Project

Table 9 – Summary of Elements Project blockchains

	Elements Project
Purpose	Create blockchain applications that utilize Bitcoin blockchain
What kind of data can be stored?	Cryptocurrency transactions
Scripting Languages	Language similar to Bitcoin, but with enhancements
Is the ecosystem open?	Alpha sidechain: Yes Sidechain you create: (unknown)
How can one participate?	Download the source code from GitHub and follow their instructions. Then join Alpha sidechain or create your own sidechain.
Native Currency	hostcoin
Who are the registration authorities?	(unknown)
Is decision making transparent?	(unknown)
Does it use a managed PKI?	(unknown)
Who manages PKI?	(unknown)
Block-release Timing	Configurable (default: 60 sec.)
Transaction Size	(unknown)
Transaction Rate	(unknown)
Consensus Model	(unknown)
Mining	(unknown)

8. Lisk

In addition to the development of sidechains through the Elements Project, the Lisk Foundation is also working on sidechain technology using the blockchain implementation they created, known as Lisk. The goal with Lisk is to enable users to create their own applications on separate sidechains that are connected to the main blockchain, or *mainchain*, of Lisk. The mainchain simply records LSK transactions between accounts. Any additional functionality must be programmed within a sidechain linked to the mainchain. According to Max Kordek, co-founder and CEO of Lisk, “This will give millions of developers the ability to create their own sidechains, particularly around consumer applications, including games, social networks, and the Internet of Things, but the same core functionality can also be used to develop and scale business applications” (5). The platform is written in JavaScript, and utilizes NodeJS and PostgreSQL in the backend. New users can participate by logging in through the main website and buying the native currency, denoted as LSK. The website also has downloads for a program to send and receive transactions on the Lisk network, and another program to enable API calls. Furthermore, the source code is available on GitHub (https://github.com/LiskHQ), along with a wiki that includes documentation about the platform.

Consensus and mining is done through a Delegated Proof-of-Stake model like the Steem network, only slightly different. The Lisk network is protected by 101 active delegates, each of which is elected by the stakeholders of LSK. All delegates are ranked by the number of votes they have received, and only the top 101 delegates are considered active. Only active delegates can generate new blocks. The consensus process contains multiple rounds. At the start of each round, the order in which delegates can forge relative to each other is determined. Then, each delegate forges exactly one block, which can include up to 25 transactions. Unlike Bitcoin, a proof-of-work is not required to create blocks. A delegate earns a

fixed amount of LSK if they successfully submit a block that is accepted by the system. In addition, participants in each round earn a portion of the transaction fees spent on transactions submitted during that round.

Blocks are created within 10 seconds, and the network can support up to 250 transactions per 10 seconds, or 25 transactions per second. The type of a transaction determines its maximum size, which can be up to 1223 bytes.

Lisk uses the Edwards-curve Digital Signature Algorithm (EdDSA) for hashing, which the developers believe is faster than the ECDSA used by Bitcoin. A user generates a private key-public key pair to participate in the network. The private key allows the user to sign transactions, while the public key is included in transactions the user sends and verifies the validity of the signature. For additional security, Lisk allows users to register a second pass phrase associated with their public key, thus requiring all subsequent transactions to be signed using the second pass phrase to be considered valid.

Since the Lisk platform is still in development, its full capabilities are still unknown. Hopefully, the protocol will progress to a point where users can create a wide variety of applications on sidechains.

8.1. Summary of Lisk

Table 10 – Summary of Lisk blockchain

	Lisk
Purpose	Allow users to create their own blockchain apps on sidechains.
What kind of data can be stored?	Mainchain: Cryptocurrency transactions. Sidechains: (unknown)
Scripting Languages	JavaScript
Is the ecosystem open?	Yes
How can one participate?	Create a new account on Lisk webpage, and buy LSK. Then create new sidechain by following instructions in documentation.
Native Currency	LSK
Who are the registration authorities?	(unknown)
Is decision making transparent?	Delegates (those who can mine currency) are known
Does it use a managed PKI?	(unknown)
Who manages PKI?	(unknown)
Block-release Timing	10 seconds
Transaction Size	1223 bytes maximum
Transaction Rate	25 tx/sec.
Consensus Model	Delegated Proof-of-Stake
Mining	Part of the consensus model; no proof-of-work

9. Summary of Networks

After reading through the previous descriptions of various blockchain implementation, one should generally understand how each blockchain functions and the pros and cons of each. The following tables summarize the important aspects of all blockchains discussed in this paper.

Table 11 – Summary of all blockchains: Bitcoin, Litecoin, Dogecoin, Ethereum

	Bitcoin	Litecoin	Dogecoin	Ethereum
Purpose	Cryptocurrency	Cryptocurrency	Cryptocurrency	Run smart contracts
What kind of data can be stored?	Cryptocurrency transactions, plus some additional data in coinbase or OP_RETURN transactions	Cryptocurrency transactions, plus some additional data in coinbase or OP_RETURN transactions	Cryptocurrency transactions, plus some additional data in coinbase or OP_RETURN transactions	Cryptocurrency, digital assets, smart contracts
Scripting Languages	Script	Script	Script	Solidity, Serpent, LLL
Is the ecosystem open?	Yes	Yes	Yes	Yes
How can one participate?	Download the source code from GitHub, and follow their instructions. Obtain currency from online trading service.	Download the source code from GitHub, and follow their instructions. Obtain currency from online trading service.	Download the source code from GitHub, and follow their instructions. Obtain currency from online trading service.	Download the source code from GitHub, and follow their instructions. Obtain currency from online trading service.
Native Currency	bitcoin (BTC)	litecoin (LTC)	Dogecoin (DOGE)	ether (ETH or ETC)
Who are the registration authorities?	N/A	N/A	N/A	N/A
Is decision making transparent?	Yes	Yes	Yes	Yes
Does it use a managed PKI?	No	No	No	No
Who manages PKI?	N/A	N/A	N/A	N/A
Block-release Timing	10 minutes	2.5 minutes	60 sec.	12 sec.
Transaction Size	200 bytes minimum, 250 bytes avg.	(unknown)	(unknown)	Theoretically no max (actual max: 89 kB)
Transaction Rate	3 transactions/sec. avg., 7 transactions/sec. theoretical maximum	28 transactions/sec. theoretical maximum	70 transactions/sec. theoretical maximum	Theoretically no maximum
Consensus Model	Nodes verify blocks and transactions, and	Nodes verify blocks and transactions, and	Nodes verify blocks and transactions, and	Similar to Bitcoin, but uses Ethereum Virtual Machine

	Bitcoin	Litecoin	Dogecoin	Ethereum
	select blockchain with the most blocks.	select blockchain with the most blocks.	select blockchain with the most blocks.	
Mining	Proof-of-work	Proof-of-work	Proof-of-work	Proof-of-work using Ethash algorithm

Table 12 – Summary of all blockchains: MultiChain, Hyperledger Fabric, Hyperledger Sawtooth, Hyperledger Iroha

	MultiChain	Hyperledger Fabric	Hyperledger Sawtooth	Hyperledger Iroha
Purpose	Provide a platform for creating your own blockchain.	Enable the creation of blockchains for industry use cases.	Enable companies to deploy their own blockchains.	Provide tools that integrate easily into existing environments.
What kind of data can be stored?	Any digital asset you want to store.	Chaincode (i.e. smart contracts)	Anything that can be defined by a "transaction family".	(unknown)
Scripting Languages	N/A	Go (golang), Java (in progress)	Python	(unknown)
Is the ecosystem open?	Configurable	No	Configurable	(unknown)
How can one participate?	Install MultiChain app, and follow online instructions to make a blockchain.	Create a blockchain: Download source and follow instructions. Join existing network: Register with a proof of identity to the network membership services.	Download the source code from GitHub and follow their instructions.	(unknown)
Native Currency	N/A	N/A	Initially provides the MarketPlace transaction family, which can track assets.	(unknown)
Who are the registration authorities?	Configurable	Defined before blockchain is initialized, and more can be assigned while running.	None, unless optional administration key is used.	(unknown)
Is decision making transparent?	Configurable	(unknown)	Yes; transaction transparency is default	(unknown)
Does it use a managed PKI?	No	Yes	No	(unknown)
Who manages PKI?	N/A	One or more entities in	N/A	(unknown)

	MultiChain	Hyperledger Fabric	Hyperledger Sawtooth	Hyperledger Iroha
		membership services.		
Block-release Timing	Configurable	(unknown)	Configurable	2 seconds
Transaction Size	Maximum size configurable	(unknown)	(unknown)	(unknown)
Transaction Rate	Configurable	> 10k transactions/sec.	(unknown)	(unknown)
Consensus Model	Fixed ratio of admins approves privilege changes. Longest valid blockchain adopted as global consensus.	Pluggable consensus framework; 2 plugins provided: PBFT, and "dummy" plugin	Provides two: Proof of Elapsed Time (PoET), and Quorum Voting	Sumeragi
Mining	Round-robin system; proof-of-work requirement is configurable	N/A	N/A	(unknown)

Table 13 – Summary of all blockchains: Steem, Elements Project, Lisk

	Steem	Elements Project	Lisk
Purpose	Social media platform that rewards users for meaningful posts.	Create blockchain applications that utilize Bitcoin blockchain	Allow users to create their own blockchain apps on sidechains.
What kind of data can be stored?	Posts (text and pictures), votes, comments, profiles, and follows	Cryptocurrency transactions	Mainchain: Cryptocurrency transactions. Sidechains: (unknown)
Scripting Languages	(unknown)	Language similar to Bitcoin, but with enhancements	JavaScript
Is the ecosystem open?	Yes	Alpha sidechain: Yes Sidechain you create: (unknown)	Yes
How can one participate?	Steemit: Make an account online and obtain assets. Blockchain: Download source code and follow their instructions.	Download the source code from GitHub and follow their instructions. Then join Alpha sidechain or create your own sidechain.	Create a new account on Lisk webpage, and buy LSK. Then create new sidechain by following instructions in documentation.

	Steem	Elements Project	Lisk
Native Currency	Steem (STEEM), Steem Power (SP), and Steem Dollars (SMD)	hostcoin	LSK
Who are the registration authorities?	N/A	(unknown)	(unknown)
Is decision making transparent?	You know who upvotes what, but forking process isn't clear.	(unknown)	Delegates (those who can mine currency) are known
Does it use a managed PKI?	No	(unknown)	(unknown)
Who manages PKI?	N/A	(unknown)	(unknown)
Block-release Timing	3 seconds	Configurable (default: 60 sec.)	10 seconds
Transaction Size	100 bytes avg.	(unknown)	1223 bytes maximum
Transaction Rate	10k transactions/sec	(unknown)	25 transactions/sec.
Consensus Model	Delegated Proof-of-Stake	(unknown)	Delegated Proof-of-Stake
Mining	Proof-of-work	(unknown)	Part of the consensus model; no proof-of-work

Conclusion

The comparison of different blockchain implementations reveals that no two networks are exactly alike. Separate networks may differ in purpose, performance, governance mechanisms, security algorithms, or some other ways. These differences can be good or bad depending on the requirements for the application utilizing a blockchain. But even now, most implementations focus on cryptocurrency transactions. While these blockchains obviously work well for monetary applications, they may not be optimal for storing other kinds of data. In these cases, it may be necessary to dive into the source code of the implementation to see how one can work around the blockchain's limitations. However, if the code is not open-source, it will be very difficult to make changes that work against the protocol's original purpose.

If none of the implementations listed is a good fit for an application, then it might be time to consider implementing a new blockchain. Building a blockchain can provide control over all factors that were considered in this paper for existing implementations. For example, it would enable the creation of new permissioned or permissionless networks, depending on the application's intended audience. It would allow modification of important performance factors, especially the block creation time. Many people dislike the fact that bitcoin has a 10-minute block creation time, so avoiding this issue is very significant. One could even implement their own PKI tied to the blockchain network so they can provide improved security in the network, and have the power to fix any problems in the infrastructure. Creating a blockchain would allow fine-tuned control over each user's permissions in the network. There may also be other factors that are important for some applications but were not considered in this paper. A blockchain created from scratch can be flexible enough to accommodate such factors. However, the robustness of the security model for a custom blockchain can be hard to assess. Moreover, leveraging open source solutions provides the potential of leveraging expertise of a community of developers akin to crowdsourcing.

The cable industry should consider leveraging blockchains to improve their work. It is currently under investigation which use cases could apply to the cable industry and utilize blockchains. According to Steve Goeringer, there are three use cases that may be a very good fit: “improving trust in content distribution, streamlining complex service delivery in the medical industry by leveraging cable, and providing improved secure digital content production and distribution” (1). There could be more applications that use blockchains and are relevant to the cable industry, but finding such applications will require more research.

Abbreviations and Definitions

1. Abbreviations and Definitions

1.1. Abbreviations

API	application programming interface
BFT	Byzantine fault tolerance
BTC	bitcoin
D.A.O.	Decentralized Autonomous Organization
DOGE	Dogecoin (the currency, not the blockchain)
DPOS	Delegated Proof-of-Stake
ECA	enrollment certificate authority
ECDSA	Elliptic Curve Digital Signature Algorithm
EdDSA	Edwards-curve Digital Signature Algorithm
ECert	enrollment certificate
EOA	externally owned account
ETC	Ethereum Classic
ETH	Ethereum
EVM	Ethereum Virtual Machine
LES	Light Ethereum Subprotocol
LLL	Low-level Lisp-like Language
LTC	litecoin
NIST	National Institute of Standards and Technology
PBFT	Practical Byzantine Fault Tolerance
PKI	public key infrastructure
PoET	Proof of Elapsed Time
RIPEMD160	RACE Integrity Primitives Evaluation Message Digest 160
SGX	Software Guard Extension
SHA256	Secure Hash Algorithm 256
SMD	Steem Dollars
SP	Steem Power
SPV	simplified payment verification
STEEM	Steem (the currency, not the blockchain)
TCA	transaction certificate authority
TCert	transaction certificate
TEE	trusted execution environment
TLS-CA	transport layer security certificate authority

1.2. Definitions

application programming interface	A set of subroutine definitions, protocols, and tools for building application software.
block-release timing	The minimum time to create a new block on a blockchain.
chaincode	Programmatic code on a blockchain, similar to a smart contract.
contract account	An account on the Ethereum blockchain containing a snippet of code that gets triggered by transactions or messages received from externally owned accounts.
Ethash	The proof-of-work algorithm for Ethereum.
Ethereum Virtual Machine	The runtime environment for smart contracts on the Ethereum blockchain.
externally owned account	An account on the Ethereum blockchain that can send transactions and is controlled by private keys.
fork	An event where there are two candidate blocks competing to be part of the longest blockchain.
hash	A digital fingerprint of some binary input.
hard fork	A change in the blockchain protocol where a subset of previously invalid blocks and/or transactions are made valid, or vice-versa.
hostcoin	The native currency of sidechains created through the Elements Project.
mainchain	The main blockchain within a blockchain network.
miner	A network node that finds valid proof-of-work for new blocks, by repeated hashing.
mining	The process of adding transaction records to a blockchain ledger.
mining diversity	A parameter in a MultiChain blockchain that sets the minimum proportion of miners required to participate in round-robin mining.
nonce	A counter used for the proof-of-work algorithm.
OP_RETURN	A script opcode used to mark a transaction output as invalid.
proof-of-stake	A type of algorithm by which a cryptocurrency blockchain network aims to achieve distributed consensus.
proof-of-work	A piece of data which is difficult to produce but easy to verify.
simplified payment verification	A method for verifying if specific transactions are included in a block without downloading the entire block.
soft fork	A change in the blockchain protocol where a subset of previously valid blocks and/or transactions are made invalid.
software fork	A copy of source code from one software package that is modified independent from the original software.
stack-based language	A programming language that relies on a stack machine model for passing parameters.

Bibliography & References

Goeringer S. A Simple Overview of Blockchains: Why They Are Important to the Cable Industry. Cable-Tec Expo 2017. 2017 Oct.

Antonopoulos A. M. Mastering Bitcoin. Sebastopol (CA): O'Reilly Media, Inc.; 2014. Chapter 4, Keys, Addresses, Wallets; p. 66.

Scott, N. Second Update to July 14 Security Announcement from Steemit CEO Ned Scott. Steemit. 2016 July 18. [accessed 2016 Sept 19]. https://steemit.com/announcement/@midou05/second-update-to-july-14-security-announcement-from-steemit-ceo-ned-scott.

Back A, Corallo M, Dashjr L, Friedenbach M, Maxwell G, Miller A, Poelstra A, Timón J, Wuille P. Enabling Blockchain Innovations with Pegged Sidechains. Blockstream. 2014 Oct 22. [accessed 2016 Aug 1]. https://blockstream.com/sidechains.pdf.

Lisk Releases First Modular Cryptocurrency with Sidechains. Bitcoin PR Buzz. 2016 May 24. [accessed 2017 July 14]. http://bitcoinprbuzz.com/lisk-releases-first-modular-cryptocurrency-with-sidechains.

