Selecting Effective Blockchain Solutions

Carsten Maple[1] and Jack Jackson[2](✉)

[1] University of Warwick, Coventry CV4 7AL, UK
[2] Everledger, London WC2H 9JQ, UK
jack.jackson@linace.ox.ac.uk

Abstract. Distributed ledger technologies (DLT) are becoming increasingly popular and seen as a panacea for a wide range of applications. However, it is clear that many organisations, and even engineers, are selecting DLT solutions without fully understanding their power or limitations. Those that make the assessment that blockchain is the best solution are provided little guidance on the vast array of types of blockchain; whether permissioned, permissionless or federated; which consensus algorithm to use; and a range of other considerations. This paper aims to addresses this gap.

Keywords: Distributed ledger technology · Blockchain anatomy
Blockchain selection · Consensus determination protocol

1 Introduction

A distributed ledger is a form of database which is stored across a number of computing devices within a network. Each node independently maintains an identical copy of the shared ledger. The distribution is typically unique, as records are independently constructed and held by each participant; as opposed to information sourcing from a central authority. Each participant in the network places a vote reflecting their calculated output, allowing the group to come to a consensus regarding the *true* data output. Once consensus has been reached, the distributed ledger can then be updated by all parties. Distributed ledger technology (DLT), affords a level of dexterity which is currently not facilitated by centralized systems. A popular form of distributed ledger technology is known as blockchain. Blockchain technologies are distinguishable from DLT by the nature in which they store information. Data on a blockchain is grouped together and organised into a series of cryptographically interconnected *blocks*. A blockchain is an append-only structure, meaning that it only permits further data contribution to the ledger. Once a block of data is committed to the blockchain, it is impossible to alter or delete the information contained within. The immutable nature of blockchain means that the technology is well suited for: records management, transaction processing and auditing. Because of this, we are seeing the widespread adoption of this technology across a number of high risk supply

https://doi.org/10.1007/978-3-030-10549-5_31

chains, where fraudsters have long taken advantage of opaqueness in the product life-cycle; typically with expensive ramifications. The nature of communications in such systems can help to eliminate intermediaries, allowing individuals and organisations to interact freely with one another. Considering that many industry infrastructures consist of such middlemen (such as distributors in supply chain networks), this could vastly decrease the operational costs associated with running a business; consequently, providing a more open, transparent and competitive marketplace. The implementation of blockchain technology is also valuable within industries that are continuously subjected to rigorous regulation, as the meeting of such requirements can be both time consuming and expensive for an organisation.

The first notable implementation of DLT came in the form of the Bitcoin whitepaper [1], written by Satoshi Nakamoto. Since then DLT such as blockchain have become a panacea for a wide range of applications. They are often perceived as the cornerstone of technological innovation across industries. As such, many organisations show interest in implementing the technology within their industry. Companies wishing to adopt such mechanisms are faced with the challenge of selecting an appropriate configuration for their functional requirements. As the solutions landscape continues to grow, it becomes increasingly important that engineers and decision-makers within an organisation fully understand the power and limitations of the various technologies on offer [7,8,13,14].

In this paper we present an anatomy of blockchain solutions, analysing their key technological features. This paper begins by considering existing work within the field, before proposing and analysing a generic anatomy. This is followed by a brief discussion of a number of developed platforms, before concluding and proposing areas for future work.

2 Existing Work

Since the rise in popularity of DLT, a number of papers have been published analysing various areas within the field, such as: consensus, scalability, mining difficulty and architecture. Pass et al. [2] conducted an analysis on the original blockchain consensus mechanism [1], proving strength of consistency within an asynchronous network environment. Their paper provided proof that Nakamoto's protocol satisfied their definitions for chain quality, growth, consistency (and the upper-bound on chain growth). More generally, Glaser [6] provided a framework for blockchain-enabled systems which was heavily grounded in hard use cases. Zheng et al. [5] extend upon this, educating readers on the key features of a blockchain product, with respects to both architecture and consensus. Their focus was directed towards the benefits of using blockchain technology, referencing a number of widely adopted consensus mechanisms. This paper builds upon such work, educating readers on the selection process, in respects to which anatomical structures are most appropriate given their functional (business) requirements. Similarly, Vukolić [3] draws contrast between traditional proof-of-work (PoW) blockchains and those grounded in byzantine-fault-tolerance (BFT) schemes. Notable emphasis is place upon the computational expense associated with each of the discussed mechanisms. The scalability constraints placed upon each approach are critically analysed, addressing a long-term problem within the DLT community.

In a more critical sense, Kraft [4] focuses on analysing the computation inefficiencies within existing blockchain technologies. In his paper, Kraft identified inefficiencies within the mining processes of both Bitcoin and Namecoin. He proposes a mechanism which is designed to work "*perfectly*" across both constant and exponentially growing hash rates. Comparisons between his mechanism and more traditional methods are drawn from three core metrics: mining difficulty, average block-time and name expiration. The majority of academic work within the field of DLT has been focused on analysing one core metric of the technology. Very little work has been directed at educating readers in the identification process for the appropriate solution for their requirements. This paper builds upon a number of previous works, allowing us to address this knowledge gap.

3 An Anatomy of Blockchain Solutions

We now consider some of the key features of a blockchain network. The feature list contained within this section does not fully encompass the topology of all blockchain solutions. Instead, they are a select subset of features used by us to create a generalized anatomy of a blockchain solution. Each block within a blockchain network contains a series of transactions, accompanied by the signature of the sending network participant. Participation within a network is controlled by the permission-determination protocol adopted by the network. Network participants are often offered incentives to maintain the ledger. Within public networks this can be achieved through mining. Consensus mechanisms are used within blockchain networks to ensure that the ledger is maintained in a valid and correct state, reflecting the *true* series of transactions taking place.

3.1 Blocks

Each block within a blockchain network contains both a header and a body. The block header typically consists of information equivalent to metadata. The header is usually used to support the identification and verification of blocks and their body contents. The block body usually contains information surrounding the transactions within the block. However, the format of data and information contained within both the header and body, varies between implementation.

For example, within the Bitcoin [1] network, the header contains information pertaining to the block version, which is an indication as to which set of validation rules it follows. The header also contains: a hash of all transactions within the block, a universal timestamp, a target threshold of a valid block hash, a 4-byte field and a 256-bit hash from its parent block. Each block has only one parent. The block body contains a series of transactions accompanied by a transaction counter. The maximum number of transactions contained within a block is dependent upon the size of both the block and the transaction contained within.

In contrast, the Ethereum [14,15] network block contains vastly different content. The block head within an Ethereum block (based on the yellow paper [15]) contains the: parent hash, ommers hash, beneficiary, state root, transactions root, receipts root, logs bloom, difficulty, number, gas limit, gas used, timestamp, extra data, mix hash and nonce. Whilst the body contains a list of transactions and an ommers list.

3.2 Smart Contracts

In essence, smart contracts [18] are a protocol which allow for the contribution, verification and implementation of actions within a contract. Smart contracts can be implemented to ensure the effectuation of embedded regulation, contractual terms, business rules and other codifiable instructions. This allows for the enforcement of obligations along with the auditable tracking of that enforcement on the blockchain, in addition to transparency and third party integration. This creates a unique value proposition for users, as well a proven platform on which to further integrate partners' systems and processes.

In the event that manual checks are required, smart contracts are capable of triggering an event which notifies relevant parties that action is necessary. This also extends to situations in which a combination of document submission and manual checking is required. When critically analysing these processes, it would be a waste of time to have a human perform an analysis if other supporting documents are either not present or invalid. A smart contract is capable of managing such scenarios, by first ensuring all supporting documents are present and valid, before prompting the human interaction with the process. Essentially, enabling effective time-management across an organisation.

While many blockchain platforms use smart contracts, such as: Ethereum [14,15], Corda [13] and Hyperledger Fabric [12], not all do; with Monero [7] being a prime example of this.

Monero is a platform which claims to offer superior security and privacy to its competitors, through the implementation of mechanisms such as ring signatures. Implementing smart contracts into a platform increases the attack surface of the network. All of which is counter-intuitive for a privacy-focused platform such as Monero.

3.3 Transaction Signing

Transaction signing is used within networks to provide assurance surrounding the validity of transactions. Networks build their transaction signing mechanisms upon a wide range of cryptographic mechanisms. One such example, is elliptic curve (EC) cryptography. Elliptic curves can be applied within asymmetric cryptography, to allow network participants to maintain ownership of their own public and private keys. Parties use their private keys to sign transactions using the elliptic curve digital signature algorithm (ECDSA). Signed transactions are then broadcast across the network for peer verification. Digital signature algorithms (DSA) such as ECDSA are used to ensure the authenticity of both the source and integrity of data. A great example of elliptic curves being implemented for transaction signing can be seen in the secp256k1 elliptic curve, which is implemented within: Bitcoin [1], Ethereum [14,15] and Zcash [8]. Numerous different types of elliptic curves are used by blockchain networks in the signing process, including the Ed25519 curve used within Monero [7].

3.4 Permissions

Permissions control access parameters around participation within a network, and can either encourage open contribution from the general population, or restrict admittance to pre-trusted entities. This is extremely useful from an engineering perspective as it provides flexibility, adaptive to the specification requirements of the proposed solution. Blockchain has three core permission-determination protocols: permissionless, permissioned and federated.

Permissionless blockchain networks [20] allow for external parties to participate within an open network, without the need for traditional registration mechanisms (permissionless access). This creates openness without imposing strong self-identification restraints on actors within the network. Since anyone can participate in the network freely, there is a requirement to offer a financial incentive for peer validation and system maintenance tasks. Incentives help to promote peer participation in the network, which in turn has a positive impact on the overall security provided by the system. For example, within the PoW consensus mechanism, a 51% network majority is required to enforce change.

Permissioned blockchains [20] are not controlled or manipulated by a single entity, but by a consortia of entities, which provide selected parties with authorization to participate within the network. This affords the consortia the power to *vet* participants to a degree. Since permissioned networks operate under the supervision of a trusted consortium of validators, the cost of verifying transactions is significantly reduced. The infrastructure also supports the updating of protocols with relative ease. As all nodes are well known and regulated, there is negligible risk of network performance degradation or outage.

Federated blockchains also restrict access to the network. However, there is a subtle difference in functionality from permissoned networks. For example, imagine that a network consisted of 15 nodes. Within a permissioned network each node would be required to sign transactions. However, within a federated network, only a subsection of the overall consensus contributing nodes are required to sign blocks. Flexibility is also afforded when assigning read permissions to the ledger. Conducting consensus in this nature reduces the computational cost associated with transaction signing and data redundancies even further.

3.5 Consensus

Consensus is the mechanism by which individuals within a network come to an agreement regarding a particular decision or view. Ideally, consensus mechanisms are capable of mitigating the actions of dishonest individuals or groups; whilst offering a level of redundancy in the event that a subset of parties are unable to respond. Within blockchain technology, consensus is used to provide a degree of consistency across transactions within a ledger. Each consensus mechanism has varying degrees of tolerance around the level of redundancy, resilience, speed and security involved in the performance of their functionality. Common consensus determination algorithms used within blockchain platforms include: PoW [1,3,19], proof-of-stake (PoS) [19], delegated proof-of-stake (DPoS), proof-of-authority (PoA), proof-of-elapsed-time (PoET) [16], byzantine fault tolerance (BFT) [3,9,11].

Proof-of-Work [1,3] is a piece of data that is both time consuming and costly to produce, but easily verifiable by peers. Producing a PoW is essentially a random process with a low probability of success; thus, it is down to trial and error. Perhaps the most notable implementation of PoW can be seen within the Bitcoin network [1]. Within Bitcoin, block references are obtained through the double hashing of the blocks header content, using the SHA-256 hash function. Hashing, mining difficulty and threshold functions for PoW schemes are outlined within [19]. Due to the nature of PoW, an adversarial party would require 51% of the networks hash-rate (or computing power) to compromise the system.

Although PoW is a decentralized protocol, within networks such as Bitcoin, PoW is executed by an ever increasing centralized system of miners. Practically speaking, most people do not have the computational resources to participate in the mining process. As a result, to guarantee stable returns for participation, most miners pool their resources. This has effectively created a centralized mechanism, revolving around so called *pool managers*. Currently, the Bitcoin network, the five biggest mining pools control over $\frac{3}{4}$ of all hashing power.

Proof-of-Stake [19] derives its consensus from the holdings of the cryptocurrency itself. For instance, if Bitcoin were to adopt PoS, users who hold the largest amount of Bitcoin would have the authority to make change across the network. Majority owners would also be able to mine an equivalent portion of their funds regardless of computing power. For an explanation of how the PoS algorithm works; including how it adjusts difficulty and provides proof of address ownership, read [19]. There is a known flaw within the PoW scheme in which newcomers to a network, without prior knowledge of the chain, can be tricked into validating an invalid chain, based on its larger length. PoS combats this by implementing a rule-set which disallows forking from a branching point more than N blocks in the past. Therefore, new participants in the network are provided with all information of prior block content. However, it must be noted that this requires a trade-off in the form of a centralization, trusted source. For a PoS mechanism to work effectively, there needs to be a way to select forgers (a transaction validator) from a user group. Simply selecting forgers based on their account balance would result in a system which is heavily skewed in favour of richer participants, who decide to stake more of their currency. To counter this problem a number of selection mechanisms have been created, such as randomized block and coin age based selection.

It is worth noting that a semi-centralized PoS algorithm known as delegated proof-of-stake also exists. Within DPoS, blocks are created by a select set of users within the system known as delegates. Stake within DPoS is used slightly differently than within PoS. For example, delegates may be *elected* based on their stake in proportion to the network. Additionally, delegates may receive votes from network participants. The voting power of participants is also reliant on their stake relative to the entire network. Delegates are rewarded for performing their role and punished for misbehaviour. There are two core functions performed by delegates: block building and signing. Each delegate is capable of individually generating blocks; however typically speaking, multiple delegates are required to sign a block. Signing is performed by a select subset of delegates. One notable exception to this rule is Tendermint [10], in which any user in the system canprovide a signature.

In essence, PoS systems are more computationally efficient than their PoW counterpart. A greater number of people are encouraged to run nodes and participate in the network as the cost of participation is affordable. As a result, they system becomes increasingly decentralized. Additional protection comes from the expense associated with executing an attack and the reduced incentive for attackers. An attacker would require a near majority of all network currency to manipulate the network. This is typically referred to as the *monopoly problem.*

Byzantine Fault Tolerance is commonly thought of in regard to the byzantine generals problem [9]. With respect to the byzantine generals problem, byzantine fault tolerance is achieved when loyal generals come to a majority agreement on their strategy. Typically byzantine faults are the most challenging to deal with, as no restrictions or assumptions are made around the behaviour a node can exhibit. The most successful approach to date is known as *practical byzantine fault tolerance* (PBFT) [11].

PBFT is the core for many algorithms used for tasks, such as: terminating reliable broadcast, group membership, view synchronous b-cast and state machine replication. PBFT processes can be categorised into three core types: clients, primary n replicas and backup n replicas. Any client within the system can be faulty. In accordance with the $1/3^{rd}$ corruption tolerance of BFT, the number of replicas present is calculated as follows; $n = 3f + 1$, where f is upper bound of faulty parties.

Practical Byzantine Fault Tolerant Consensus [11] is achieved through a three-phase protocol, ensuring the validity and integrity of the agreement. The first phase is known as pre-preparation. Within this phase an order of requests within the same view is agreed upon. The preparation phase that follows again agrees upon a request order within the same view, whilst also ensuring that request execution is performed in the pre-prepared order across different views. Garbage collection is also performed as part of the preparation phase. Finally a commit phase is executed, which again ensures the order of request execution in accordance with the preparation phase, whilst handling further garbage collection duties.

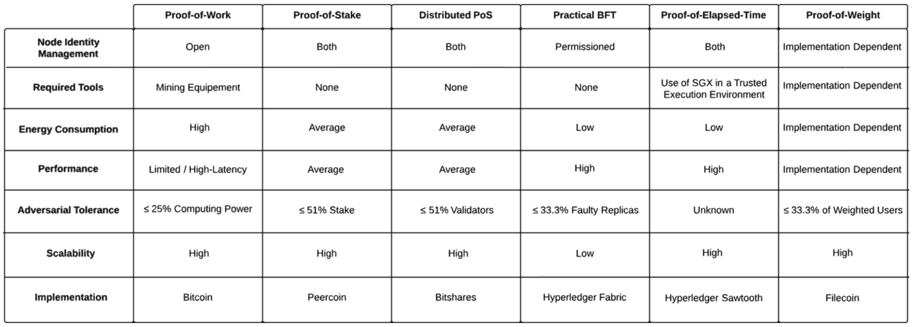
PBFT consumes less energy that both PoW and PoS mechanisms and also offers a higher level of performance in respects to latency and network throughput. Since it is a permissioned network protocol, its adversarial tolerance should not be compared to that of PoW, or permissionless PoS schemes. However, due to the nature of PBFT consensus, scalability becomes a significant issue. Therefore, we recommend using PBFT within small to medium size networks. PBFT is particularly effective within industries consisting of a strictly defined infrastructure.

Proof-of-Elapsed-Time (PoET) [16] is designed to achieve a distributed consensus in a *lottery-type* function. It was originally designed with the aim of creating a fair mechanism for distributing mining rights within permissioned networks.

PoET abide by a four-stage process flow. Firstly, each validator requests a randomly distributed wait-time period from a trusted enclave. Secondly, the validator with the shortest wait time wins the election and is awarded leadership for the transactional block in question. A function is used to create a timer for the transaction block that is guaranteed to have been created by the enclave. Another function is then used to verify the timer origin.

The enclave comes in the form of a secure CPU instruction-set, ensuring fairness in the randomness of selection across all participants. This is achieved through implementing in the low level, using Intel's Software Guard Extensions (SGX) [17]. This facilitates the PoET algorithm in providing random distribution of leadership across an entire population of participants, in a fair manner. An attestation of execution provides verification of a participants claim to leadership, providing a low cost for participation. This offers a strong incentive for participation in the network, as the algorithm is perceived to be fair, just and accessible (due to the low cost infrastructure). Perhaps the most notable implementation of PoET can be seen within the Hyperledger Sawtooth [16] network.

The following diagram provides a comparison between the aforementioned consensus mechanisms Fig. 1.



	Proof-of-Work	Proof-of-Stake	Distributed PoS	Practical BFT	Proof-of-Elapsed-Time	Proof-of-Weight
Node Identity Management	Open	Both	Both	Permissioned	Both	Implementation Dependent
Required Tools	Mining Equipement	None	None	None	Use of SGX in a Trusted Execution Environment	Implementation Dependent
Energy Consumption	High	Average	Average	Low	Low	Implementation Dependent
Performance	Limited / High-Latency	Average	Average	High	High	Implementation Dependent
Adversarial Tolerance	≤ 25% Computing Power	≤ 51% Stake	≤ 51% Validators	≤ 33.3% Faulty Replicas	Unknown	≤ 33.3% of Weighted Users
Scalability	High	High	High	Low	High	High
Implementation	Bitcoin	Peercoin	Bitshares	Hyperledger Fabric	Hyperledger Sawtooth	Filecoin

Fig. 1. Consensus protocol comparison

4 Developed Platforms

Once an engineer has determined that a blockchain solution is required it is then necessary for them to undertake the platform selection process. We discuss some of the key platforms here.

Corda [13] is an open-source DLT, which is targeted towards the financial industry. Corda is a permissioned private platform, offering a plugable consensus mechanism on a transactional level. This affords great flexibility to an engineer when developing an application built on Corda. The nature of consensus within the system means that no global broadcasting of data occurs. This is an attractive quality within heavily regulated industries, where information secrecy is pivotal. Corda has recently been endorsed by the famous insurance consrtium b3i. Smart contracts within Corda are grounded in legal prose; allowing for the self-execution and automation of event-driven, legally binding agreements between parties. This makes Corda ideal for financial records management and the automation of financial agreements.

Ethereum [14,15] is a decentralized platform which affords developers the ability to execute smart contracts within custom built blockchain networks. Ethereum allows for the creation of cryptocurrencies and storing of crypto-assets within the Ethereum Wallet. The wallet also allows for the writing, deployment and use of smart contracts. Perhaps the most interesting use case for Ethereum, can be seen within digital identity management systems. One great example of this is uPort, which aims to give users a more secure way to provide proof of their identity, by only offering critical information required to perform the desired function. For example, only providing an airport with the relevent information when boarding an aircraft. Ethereum is ideal for such a use-case as it has an easy-to-use smart contract mechanism.

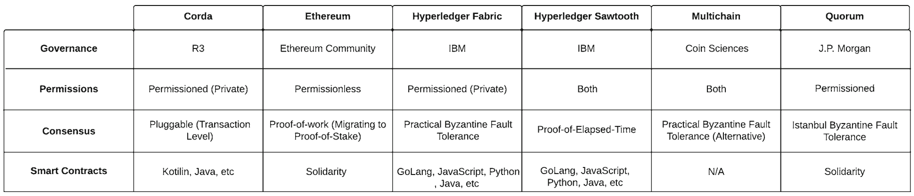
Hyperledger Fabric [12] is an open source, modular platform, currently spearheaded by IBM (previously governed by the Linux Foundation). Fabric is a private permissioned platform, which supports the use of one or more networks. Each network manages the requirements of a different set of member nodes. Fabric offers users the ability to perform queries and updates to the ledger, through the use of a series of industry standard data store mechanisms, such as: key-based lookups, composite key queries and range-based searches. Fabric utilizes PBFT [11] and conducts consensus on a transactional level. Peers within Fabric networks are required to endorse transactions in accordance with a number of predefined policies. For a transaction within the network to be validated it must pass all policy checks and receive the signature of all endorsing peers submitting to the Fabric *ordering service.*

Fabric consists of channels, which contain the configuration block which defines policies, access controls and other information important to the blockchains function. Fabric channels allow for the derivation of cryptographic materials from multiple sources. An example of a real world implementation of Fabric can be seen within the Everledger organisation, in which Fabric is used to track and trace diamonds across the supply chain network.

Hyperledger Sawtooth [16] is also part of the Hyperledger family. However, it differs vastly from its sister Fabric network. Where Fabric was built for vast business networks, Sawtooth was developed as solution aimed at reducing the computational resource consumption within large distributed validator populations. It achieves this through implemented the proof-of-elapsed-time (PoET) consensus protocol. Since consensus is performed within the CPU, which are contained within most consumer electronics, realistically speaking, almost any device can participate within consensus. Considering that PoET consensus is also exceptionally lightweight, this makes Sawtooth ideal for use cases in which IoT devices (which typically contain limited computing power) are implemented. Sawtooth is a great choice for supply-chain networks with IoT device tracking during transit.

MultiChain [21] offers a platform for building both permissioned and permissionless blockchain networks. Multichain differentiates itself from most other blockchain networks through its use of *streams*, which come in three core formats: A NoSQL key-value database or document store, an ordered time series database, or an identity-driven database with author-based entry classification. If the purpose of the blockchain product is to store information as opposed to function execution, MultiChain provides a fast and lightweight solution. For this reason, MultiChain is particularly useful for document storage systems, in which simple CRUD functionality is required. Development on the MultiChain platform is also exceptionally easy. Streams can be created and added to the network without the need to write code.

Quorum [22], similarly to Corda, is a financial service facing blockchain network. However, unlike Corda, Quorum is actually a fork of an existing blockchain platform, Ethereum. The core changes Quorum makes within its fork, is the addition of a different consensus protocol, encrypted storage and the change to a permissioned access structure. Quorum still provides access to the standard Ethereum features, such as a distributed ledger and smart contracts. Quorum solves two major existing problems preventing permissioned network implementation upon the Ethereum platform. Firstly, within Ethereum anyone can connect to the network due to its permissionless nature. Secondly, all data inside of the smart contracts within Ethereum are visible to all participant nodes. Quorum addresses these issues by taking an off-chain approach to data storage. Quorum is particularly useful within use-cases in which privacy and security of transactions are a core concern (Fig. 2).



	Corda	Ethereum	Hyperledger Fabric	Hyperledger Sawtooth	Multichain	Quorum
Governance	R3	Ethereum Community	IBM	IBM	Coin Sciences	J.P. Morgan
Permissions	Permissioned (Private)	Permissionless	Permissioned (Private)	Both	Both	Permissioned
Consensus	Pluggable (Transaction Level)	Proof-of-work (Migrating to Proof-of-Stake)	Practical Byzantine Fault Tolerance	Proof-of-Elapsed-Time	Practical Byzantine Fault Tolerance (Alternative)	Istanbul Byzantine Fault Tolerance
Smart Contracts	Kotilin, Java, etc	Solidarity	GoLang, JavaScript, Python , Java, etc	GoLang, JavaScript, Python, Java, etc	N/A	Solidarity

Fig. 2. Platform comparison

5 Conclusion

In this paper we propose a format for outlining a generic blockchain anatomy. This anatomy ranges from permissions to consensus and can be referenced when assessing blockchain solutions architecture; assist in the design and implementation of business logic within the technology. We draw comparisons between existing technologies and protocols, providing solutions architects with a baseline upon which to build their products. However, this paper is meant to be used as a guideline, and is by no means a *bible* for building blockchain solutions. From a practical perspective, within industry, it is necessary to consider a multitude of factors outside that of the technological functionality; for example, cost.

In future work, we aim to address a number of other key topics for consideration when building ontop of blockchain technologies, such as: off-chain storage requirements, privacy preservation, integration with existing systems (particularly legacy) and ease of use. This should provide engineers with further information, upon which to effectively build their solutions. Platforms such as Monero [7] which focus heavily on security would make for an interesting starting point for such work.

References

1. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf
2. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks (2016). https://pdfs.semanticscholar.org/161c/24b98ce3af2c0f8a5e96d5055a367b81801e.pdf
3. Vukolić, M.: The quest for scalable blockchain fabric: proof-of-work vs. BFT replication (2015). https://allquantor.at/blockchainbib/pdf/vukolic2015quest.pdf
4. Kraft, D.: Difficulty control for blockchain-based consensus systems (2015). https://allquantor.at/blockchainbib/pdf/kraft2016difficulty.pdf
5. Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H.: An overview of blockchain technology: architecture, consensus, and future trends (2017). https://www.researchgate.net/profile/Hong-Ning_Dai/publication/318131748_An_Overview_of_Blockchain_Technology_Architecture_Consensus_and_Future_Trends/links/59d71faa458515db19c915a1/An-Overview-of-Blockchain-Technology-Architecture-Consensus-and-Future-Trends.pdf
6. Glaser, F.: Pervasive decentralisation of digital infrastructures: a framework for blockchain enabled system and use case analysis (2017). https://scholarspace.manoa.hawaii.edu/bitstream/10125/41339/1/paper0190.pdf
7. van Saberhagen, N.: CryptoNote v 2.0 (2013). https://cryptonote.org/whitepaper.pdf
8. Ben-Sasson, E., et al.: Zerocash: decentralized anonymous payments from bitcoin (extended version) (2014). http://zerocash-project.org/media/pdf/zerocash-extended-20140518.pdf
9. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem (1982). https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf
10. Kwon, J.: Tendermint: consensus without mining (2014). https://tendermint.com/static/docs/tendermint.pdf

11. Castro, M., Liskov, B.: Practical Byzantine fault tolerance (1999). http://pmg.csail.mit.edu/papers/osdi99.pdf
12. Cachin, C.: Architecture of the hyperledger blockchain fabric (2016). https://pdfs.semanticscholar.org/f852/c5f3fe649f8a17ded391df0796677a59927f.pdf
13. Hearn, M.: Corda: a distributed ledger (2016). https://docs.corda.net/_static/corda-technical-whitepaper.pdf
14. Ethereum Contributors. "White Paper" (2018). https://github.com/ethereum/wiki/wiki/White-Paper/3592dda1feca69cce8a9d9a624ea33b0999e1dcc
15. Ethereum Community. "Ethereum Yellow Paper" (2018). https://github.com/ethereum/yellowpaper/blob/e653258d1f94c6a29ff1c2c5b43e191f535fba49/README.md
16. Intel Corporation 2015–2017. https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture.html
17. Costan, V., Devadas, S.: Intel SGX Explained (2016). https://eprint.iacr.org/2016/086.pdf
18. Clack, C.D., Bakshi, V.A., Braine, L.: Smart contract templates: foundations, design landscape and research directions (2016). http://arxiv.org/abs/1608.00771
19. BitFury Group. Proof of Stake versus Proof of Work (2015). https://bitfury.com/content/downloads/pos-vs-pow-1.0.2.pdf
20. Yaga, D., Mell, P., Roby, N., Scarfone, K.: Blockchain technology overview (2018). http://img1.wsimg.com/blobby/go/60231649-12ce-4835-96f0-945ea7f2116c/downloads/1cb8a20ea_182905.pdf
21. Greenspan, G., Founder and CEO, Coin Sciences Ltd.: MultiChain private blockchain - White Paper (2015). https://www.multichain.com/download/MultiChain-White-Paper.pdf
22. Morgan Chase, J.P.: Quorum overview (2017). https://github.com/jpmorganchase/quorum/wiki/Quorum-Overview/0600693ce1453c7513c59154a2512c1efc2044eb