A Taxonomy of Blockchain-Based Systems for Architecture Design

Xiwei Xu*†, Ingo Weber*†, Mark Staples*†, Liming Zhu*†, Jan Bosch¶, Len Bass‡, Cesare Pautasso§, Paul Rimba*
*Data61, CSIRO, Sydney, Australia
{firstname.lastname}@data61.csiro.au
†School of Computer Science and Engineering, UNSW, Sydney, Australia
‡Carnegie Mellon University, Pittsburgh, USA
lenbass@cmu.edu
§USI Lugano, Switzerland
c.pautasso@ieee.org
¶Chalmers University of Technology, Gothenburg, Sweden
jan@janbosch.com

***Abstract*—Blockchain is an emerging technology for decentralised and transactional data sharing across a large network of untrusted participants. It enables new forms of distributed software architectures, where agreement on shared states can be established without trusting a central integration point. A major difficulty for architects designing applications based on blockchain is that the technology has many configurations and variants. Since blockchains are at an early stage, there is little product data or reliable technology evaluation available to compare different blockchains. In this paper, we propose how to classify and compare blockchains and blockchain-based systems to assist with the design and assessment of their impact on software architectures. Our taxonomy captures major architectural characteristics of blockchains and the impact of their principal design decisions. This taxonomy is intended to help with important architectural considerations about the performance and quality attributes of blockchain-based systems.**

***Index Terms*—Software architecture, Distributed databases**

I. Introduction

Blockchain is the technology behind Bitcoin [16], which is a digital currency based on a peer-to-peer network and cryptographic tools. Bitcoin network provides a “trust-less” environment, where users can transfer money to each other without relying on central trusted authorities, like bank systems or payment services. A blockchain provides a kind of append-only data store of transactions replicated between peers.

Many banks are involved in trials of blockchain technology, including through the global R3 consortium[1], which is applying blockchain to trade finance and cross-border payments. Financial transactions are the first, but not the only use case being investigated for blockchain technology. A blockchain implements a distributed ledger, which can in general verify and store any kind of transactions [22]. Many startups, enterprises, and governments [1] are exploring its applications in areas as diverse as supply chain, electronic health records, voting, energy supply, ownership management, and protecting critical civil infrastructure. The wide array of interest in blockchain technology is underlined by the fast evolution of its ecosystem, including easier deployment through Blockchain-as-a-Service from Microsoft Azure[2] and IBM[3]. Blockchain has become a publicly-available infrastructure for building decentralised applications and achieving interoperability.

From a software architecture perspective, blockchain enables new forms of distributed software architectures, where agreement on shared state for decentralised and transactional data can be established across a large network of untrusted participants. This circumvents the need to rely on a central, trusted integration point which has the power to control and manipulate the system, and is a single point of failure. Applications built on blockchains can take advantage of properties such as data immutability, integrity, fair access, transparency, and non-repudiation of transactions.

However, blockchains have technical limitations. Privacy is impacted because information on a blockchain is available to all participants. For throughput scalability, mainstream public blockchains can only handle on average 3-20 transactions per second[4], whereas mainstream payment services, like VISA, can handle an average of 1,700 transactions per second[5]. Blockchain, as a software connector, has a complex internal structure and has many configurations and variants [28]. Since the advent of Bitcoin in 2008 [16], a diverse range of blockchains has emerged.

Thus, blockchains cannot by themselves meet the requirements for all usage scenarios, *e.g.*, those that require real-time processing. When building applications based on blockchains, we need to systematically consider the features and configurations of blockchains and assess their impact on quality attributes for the overall systems. In practice, the lack of reliable technology evaluation resources makes the comparison very

[1] http://www.r3cev.com/

[2] https://azure.microsoft.com/en-us/solutions/blockchain/

[3] http://www.ibm.com/blockchain/

[4] https://blog.ethereum.org/2016/01/15/privacy-on-the-blockchain/

[5] https://usa.visa.com/run-your-business/small-business-tools/retail.html

difficult.

In this paper, we propose a taxonomy that captures major architecturally-relevant characteristics of various blockchains, and indicate their support for various quality attributes. This is intended to be a basis for architecting blockchain-based systems. During design, the taxonomy may highlight trade-offs arising from design decisions related to blockchain platforms. The taxonomy is informed by existing industrial products, technical forums, academic literature and our own experience of using blockchains and developing prototypes.

The paper first gives background on blockchains in Section II. Section III describes the development and details of our taxonomy. A conceptual model for the design of blockchain-based systems is proposed in Section IV. Section V concludes.

II. Background on Blockchain

The term "blockchain" is used to refer to a data structure and occasionally to a network or system. As a data structure, a blockchain is an ordered list of blocks, where each block contains a small (possibly empty) list of transactions. Each block in a blockchain is "chained" back to the previous block, by containing a hash of the representation of the previous block. Thus historical transactions in the blockchain may not be deleted or altered without invalidating the chain of hashes. Combined with computational constraints and incentive schemes on the creation of blocks, this can in practice prevent tampering and revision of information stored in the blockchain.

The first generation of blockchains, like Bitcoin, provided a public ledger to store cryptographically-signed financial transactions [20]. There was very limited capability to support programmable transactions, and only very small pieces of auxiliary data could be embedded in the transactions to serve other purposes, such as representing digital assets (*e.g.*, document notarisation) or physical assets (*e.g.*, diamonds). The second generation of blockchains provides a general-purpose programmable infrastructure with a public ledger that records the computational results. Programs can be deployed and run on a blockchain, and are known as *smart contracts* [18]. Smart contracts can express triggers, conditions and business logic [25] to enable more complex programmable transaction. However, they are not necessarily smart, nor necessarily related to legal contracts. A common simple example of a smart contract-enabled service is escrow, which can hold funds until the obligations defined in the smart contract have been fulfilled. Ethereum[6] is the most widely-used blockchain that supports general-purpose (Turing-complete) smart contracts.

Public key cryptography and digital signatures are normally used to identify accounts and to ensure authorization of transactions initiated on a blockchain. Transactions are data packages that store parameters (such as monetary value in the case of Bitcoin) and results of function calls (such as from smart contracts). The integrity of a transaction is checked by algorithmic rules and cryptographic techniques. A transaction is signed by its initiator, to authorise the expenditure of their money, to authorise the data payload of a transaction, or the creation and execution of a smart contract.

A signed transaction is sent to a node connected to the blockchain network, which validates the transaction. If the transaction is valid and previously unknown to the node, the node propagates it to other nodes in the network, which also validate the transaction and propagate it to their peers, until the transaction reaches all nodes in the network. In a global network, this can take seconds.

Mining is the process of appending new blocks to the blockchain data structure. A blockchain network relies on *miners* to aggregate valid transactions into blocks and append them to the blockchain. New blocks broadcast across the whole network, so that each node holds a replica of the whole data structure. The whole network aims to reach a consensus about the latest block to be included into the blockchain. There are different consensus mechanisms, *e.g.*, "proof-of-work" or "proof-of-stake" (see Section III). Depending on the consensus mechanism and the required guarantees, there can be different notions of when a transaction is taken to be *committed* or *confirmed* and thus immutable.

A. Fundamental Properties

If data is contained in a committed transaction, it will eventually become in practice *immutable*. The immutable chain of cryptographically-signed historical transactions provides *non-repudiation* of the stored data. Cryptographic tools also support data *integrity*, the public access provides data *transparency*, and *equal rights* allows every participant the same ability to access and manipulate the blockchain. These rights can be weighted by the compute power or stake owned by the miner. A distributed consensus mechanism governs addition of new items; it consists of the rules for validating and broadcasting transactions and blocks, resolving conflicts, and the incentive scheme. The consensus ensures all stored transactions are valid, and that each valid transaction is added only once.

Trust in the blockchain is achieved from the interactions between nodes within the network. The participants of blockchain network rely on the blockchain network itself rather than relying on trusted third-party organisations to facilitate transactions. These five properties (immutability, non-repudiation, integrity, transparency, and equal rights) are the main properties supported in existing blockchains.

B. Other Non-Functional Properties

Data privacy and *scalability* are two points of criticism of public blockchains. As discussed earlier, in this setting privacy is limited: there are no privileged users, and every participant can join the network to access all the information on blockchain and validate new transactions. There are scalability limits on (i) the size of the data on blockchain, (ii) the transaction processing rate, and (iii) the latency of data transmission. The number of transactions included in each block is also limited by the bandwidth of nodes participating in leader election (for Bitcoin the current bandwidth per block is 1MB) [3]. Latency between submission and confirmation that a transaction has

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

been included on a blockchain is affected by the consensus protocol. This is around 1 hour (10-minute block interval with 6-block confirmation) on Bitcoin, and around 3 minutes (14-second block interval with 12-block confirmation) on Ethereum.

III. Design Taxonomy

Taxonomies have been used in the software architecture community to understand existing technologies [14], [11]. Creating a taxonomy requires classifying the existing work into a compact framework, which allows an architect to explore the conceptual design space and enables rigorous comparison and evaluation of different design options.

The design taxonomy we present here defines dimensions and categories for classifying blockchains and ways of using them in systems. It is intended to help software architects evaluate and compare blockchains, and to enable research into architectural decision-making frameworks for blockchains and blockchain-based systems.

Our taxonomy is informed by the academic literature (*e.g.*, [3], [5], [19], [7], [13], [27]), books (*e.g.*, [20]), government and technical reports (*e.g.*, [1], [2]), documents of industrial blockchain products (*e.g.*, [26]), developer forums and wikis, our investigation for the Australian government of the use of blockchains in various use cases, and our experience from implementing proof-of-concept blockchain-based systems [28]. Our discussion of architectural design issues for blockchain-based systems is structured in four sections below: the level of (de)centralisation, support for client storage and computation, blockchain infrastructural configuration, and other issues.

A. Architectural Design Regarding Decentralisation

Decentralisation devolves responsibility and capability from a central location or authority. In a centralised system, all users rely on a central authority to mediate transactions. For example in a bank, customers rely on the bank's systems to correctly adjust their account balances after a bank transfer. A central authority can manipulate the whole system, including by directly updating backend databases, or by upgrading the software that implements the system. Thus, a central authority is a single point of failure for a centralised system. In contrast, a fully decentralised currency system like Bitcoin allows people to reach agreement on who owns what without having to trust each other or a separate third party. Such a system is highly available since every full node in Bitcoin network downloads every block and transaction, checks them against Bitcoin's core consensus rules and provides the required functionality to process transactions. There are currently 5000+ nodes in the Bitcoin network[7], although not all are full nodes that form the backbone of Bitcoin. Table I represents a spectrum of (de)centralisation, from full centralisation to full decentralisation. In a system it is possible that some components or functions are decentralised while others are centralised.

There are two types of centralised systems. In the first there is a monopoly service provider, including governments and courts within a jurisdiction, and business monopolies. In the other type there are alternative providers, such as banks, online payments, or cloud computing providers. Any centralised system is a single point of failure for its users. However, where there are alternative providers, the failure of a single service provider only affects its users, and users may switch providers or use multiple providers.

At the other end of the spectrum, fully decentralised systems include permission-less public blockchains, such as the Bitcoin and Ethereum blockchains. Permission-less public blockchains are completely open: new users can at any time join the network, validate transactions, and mine blocks. Decentralised systems using anonymous validators need to protect against Sybil attacks, where attackers create many hostile anonymous nodes. Bitcoin partly guards against this using its proof-of-work mechanism, so that it is not the total number of nodes that is important for integrity, but rather the total amount of computational power. While it is easy for an attacker to create anonymous nodes, it is not easy for them to amass large amounts of computational power. A decentralised system can be defeated unless there is a majority of authority (nodes, computational power, or stakeholding). Game-theoretic attacks can change this threshold, requiring a higher (*e.g.* 66%) majority to maintain integrity [9].

There is a spectrum of possibilities between centralisation and decentralisation. Here we discuss two options for partial decentralisation: *permission* and *verification*.

1) Permission: Instead of anonymous public participation, a blockchain may be permissioned in requiring that one or more authorities act as a gate for participation. This may include permission to join the network (and thus read information from the blockchain), permission to initiate transactions, or permission to mine. Some permissioned blockchains, *e.g.*, Multichain[8], allow more fine-grained permissions, such as the permission to create assets. Permissioned blockchain networks include Ripple[9] or Eris[10]. However, the code for public blockchains can also be deployed on private networks to create a kind of permissioned blockchain using network access controls. Permissioned blockchains may be more suitable in regulated industries. For example banks are required to establish the real-world identity of transacting parties to satisfy Know-Your-Customer (KYC) regulation. Permission information can be stored either on-chain or off-chain, and permissioned blockchains may be able to better control access to off-chain information about real-world assets [21]. However, a transaction on a permission-less blockchain across jurisdictional boundaries can circumvent this and undermine regulatory controls.

There are often trade-offs between permissioned and permission-less blockchains including transaction processing rate, cost, censorship-resistance, reversibility, finality [21] and the flexibility in changing and optimising the network rules. The suitability of a permissioned blockchain may also depend on the

[7] https://bitnodes.21.co/nodes/

[8] http://www.multichain.com/

[9] https://ripple.com/

[10] https://monax.io

Table I: Blockchain-related design decisions regarding (de)centralisation, with an indication of their relative impact on quality properties (⊕: Less favourable, ⊕⊕: Neutral, ⊕⊕⊕: More favourable)

Design Decision	**Option**	**Impact**			
		Fundamental properties	**Cost efficiency**	**Performance**	**#Failure points**
Fully Centralised	Services with a single provider (*e.g.*, governments, courts)	⊕	⊕⊕⊕	⊕⊕⊕	1
	Services with alternative providers (*e.g.*, banking, online payments, cloud services)				
Partially Centralised & Partially Decentralised	Permissioned blockchain with permissions for fine-grained operations on the transaction level (*e.g.*, permission to create assets)	⊕⊕	⊕⊕	⊕⊕	*
	Permissioned blockchain with permissioned miners (write), but permission-less normal nodes (read)				
Fully Decentralised	Permission-less blockchain	⊕⊕⊕	⊕	⊕	Majority (nodes, power, stake)
		Fundamental properties	**Cost efficiency**	**Performance**	**#Failure points**
Verifier	Single verifier trusted by the network (external verifier signs valid transactions; internal verifier uses previously-injected external state)	⊕⊕	⊕⊕	⊕⊕	1
	M-of-N verifier trusted by the network	⊕⊕⊕	⊕	⊕	M
	Ad hoc verifier trusted by the participants involved	⊕	⊕⊕⊕	⊕⊕	1 (per ad hoc choice)

size of the network. Nonetheless, the permission management mechanism may itself become a potential single point of failure, not just from an operational perspective but also from a business perspective.

2) Verification: The execution environment of a blockchain is self-contained. It can only access information present in a transaction or in the transaction history of the blockchain, and the states of external systems are not directly accessible. To address this limitation, a *verifier* role can be introduced to evaluate conditions that cannot be expressed in a smart contract running within the blockchain network. A verifier is a third party that is trusted to provide some types of information about the external world. When validation of a transaction depends on external state, the verifier is requested to check the external state and to provide the result to the validator (miner), which then validates the condition. A verifier can be implemented as a server outside the blockchain, and has the permission to sign transactions using its own key pair on-demand. The concept of an *oracle* in Bitcoin is an instance of a verifier[11]. The verifier can be also implemented inside a blockchain network as a smart contract with external state being injected into the verifier periodically.

A centralised verifier becomes a potential single point of failure for the transactions relying on the verifier. To decentralise the verifier, a *distributed verifier* can be introduced. A distributed verifier is comprised of several verifiers that provide the same functionality to check the external state. All the verifiers are trusted by the whole network. In this case, a transaction that relies on external state can use a multi-signature (M-of-N) schema that requires keys from M out of N verifiers to authorise a transaction. For example, Orisi[12] on Bitcoin runs a set of independent verifiers. Orisi allows the participants involved in a smart contract to select a set of verifiers and define the value of M before initiating a conditional transaction.

In addition, participants who wish to transact with each other on a blockchain could rely on an ad hoc trusted arbitrator to resolve disputes or check external status. An arbitrator may be a human with a blockchain account that is able to sign the transaction after some validation. Alternatively an arbitrator may be automated and validate the transaction based on status taken from the blockchain and the external world. For example, Gnosis[13] is a decentralised prediction market that allows users to choose any verifier they trust, such as another user or a web service, *e.g.*, for weather forecasts.

B. Architectural Design Regarding Storage and Computation

While blockchains provide some unique properties, the amount of computational power and data storage space available on a blockchain network remains limited. In addition, using public blockchains costs real money, with a different kind of cost model than conventional software systems. In regards to cost efficiency, performance, and flexibility, major design decisions in using a blockchain include choosing what data and computation should be placed on-chain and what should be kept off-chain [28]. Table II captures some of these options, which are described in more detail below.

1) Item data: A common practice for data management in blockchain-based systems is to store raw data off-chain, and to store on-chain just meta-data, small critical data, and hashes of

[11] https://en.bitcoin.it/wiki/Contract#Example_4:_Using_external_state

[12] http://orisi.org/

[13] https://groupgnosis.com/

Table II: Blockchain-related design decisions regarding storage and computation, with an indication of their relative impact on quality properties (⊕: Least favourable, ⊕⊕: Less favourable, ⊕⊕⊕: More favourable, ⊕⊕⊕⊕: Most favourable)

Design Decision		Option	Fundamental properties	Impact		
				Cost efficiency	Performance	Flexibility
Item data	On-chain	Embedded in transaction (Bitcoin)	⊕⊕⊕⊕	⊕	⊕	⊕⊕
		Embedded in transaction (Public Ethereum)		⊕⊕⊕⊕	⊕	⊕⊕⊕
		Smart contract variable (Public Ethereum)		⊕⊕	⊕⊕⊕	⊕
		Smart contract log event (Public Ethereum)		⊕⊕⊕	⊕⊕	⊕⊕
	Off-chain	Private / Third party cloud	⊕	~KB Negligible	⊕⊕⊕⊕	⊕⊕⊕⊕
		Peer-to-Peer system		⊕⊕⊕⊕	⊕⊕⊕	⊕⊕⊕
Item collection	On-chain	Smart contract	⊕⊕⊕⊕	⊕⊕⊕⊕ (public)	⊕⊕⊕⊕	⊕
		Separate chain		⊕ (public)	⊕	⊕⊕⊕⊕
Computation	On-chain	Transaction constraints	⊕⊕⊕⊕	⊕	⊕	⊕
		Smart contract				
	Off-chain	Private / Third party cloud	⊕	⊕⊕⊕⊕	⊕⊕⊕⊕	⊕⊕⊕⊕

the raw data. However, the applications of storing item data on blockchain are not just for integration with external data. There are various uses for wholly on-chain auxiliary data, including *e.g.* "coloured coins" which are a class of overlays on Bitcoin to represent and manage real world assets.

In the Bitcoin blockchain, before *OP_RETURN*[14] was made a valid *opcode* (*i.e.*, function of Bitcoin Script language) to store arbitrary bytes in an unspendable transaction, users were able to include limited information into transactions on-chain using one of four methods. These were: writing in a coinbase transaction which is only editable by miners[15], using the *nSequence* field, using a fake account address, or using unreachable script code defined through *if* and *else* conditions. All four methods are deprecated now that *OP_RETURN* has been introduced as an official way to embed arbitrary data in a Bitcoin transaction.

Table II compares the *OP_RETURN* mechanism with other options provided by public Ethereum to store arbitrary data. There are trade-offs in cost efficiency, performance and flexibility. The *OP_RETURN* instruction returns immediately with an error so that the included data is not interpreted as a script. The default Bitcoin client only relayed *OP_RETURN* transactions up to 80 bytes, which was reduced to 40 bytes in February 2014[16]. Storing 80 bytes of arbitrary data on the Bitcoin blockchain costs roughly US$0.036[17]. It is debatable whether Bitcoin should be used to record arbitrary data.

Ethereum, on the other hand, theoretically allows storing arbitrary structured data of any size. According to the cost model given in the Ethereum yellow paper [26], every transaction has a fixed cost of 21,000 gas (gas is the internal pricing for executing a transaction or storing data), and every non-zero byte of data costs additional 68 gas. Thus, the total cost of storing 80 bytes of data on Ethereum blockchain by submitting a transaction is 26,440 gas (assuming all bytes are non-zero), which is roughly US$0.007[18].

Ethereum provides two other ways to store arbitrary data in smart contracts. For 32 bytes of data, the first option is to store the data as a variable in a smart contract (all simple types in Solidity, the script language on Ethereum, are 32 bytes). The cost of storing data in the contract storage is based on the number of *SSTORE* operation on the contract variable. In the case of storing 32 bytes, there is one *SSTORE* operation that changes the data from zero to non-zero, which costs 20,000 gas. As aforementioned, the transaction as the carrier costs 21,000 gas. The data payload of the transaction including the function signature and the actual data costs extra gas. Other than these two costs, there is a cost for creating the smart contract depending on its complexity. In total, the cost is larger than US$0.010 ($20{,}000 + 21{,}000 + 32 \times 68$ gas).

The second option is to store arbitrary data as a log event. This follows different rules for calculating cost. Logged data is stored in log topics which cost 375 gas, and where every byte of data in a log topic costs an extra 8 gas. Including the fixed cost of the carrier transaction with data payload, the rough cost of using a log event to store 32 bytes of data is US$0.005 ($21{,}000 + 375 + 32 \times 8$ gas). Storing data as a variable in a smart contract is more efficient to manipulate, but less flexible due to the constraints of the Solidity language on the value types and length. The flexibility and performance of using smart contract log events is intermediate because log events allow up to three parameters to be queried.

Finally, we note that data storage on blockchain has a different cost model than conventional data storage. Although it may seem more expensive, storing data on blockchain is

[14] https://bitcoinfoundation.org/core-development-update-5/

[15] https://en.bitcoin.it/wiki/Genesis_block

[16] https://github.com/bitcoin/bitcoin/pull/3737

[17] Assuming a typical Bitcoin transaction with one input and one output, which has about 220 bytes, the default transaction fee rate of 2×10^{-4} BTC/KB (see https://en.bitcoin.it/wiki/Transaction_fees), and with an exchange rate of US$600/BTC (see https://poloniex.com/exchange#usdt_btc),

[18] Assuming from here on a gas price of 2.5×10^{-8} Ether (see https://etherscan.io/chart/gasprice) and given an exchange rate of US$10/Ether (see https://poloniex.com/exchange#usdt_eth).

a one-time cost for permanent storage. (However, note that Ethereum allows a partial refund on reclaimed smart contract variable storage.)

Selection of off-chain data storage concerns the interaction between the blockchain and the off-chain data storage. The off-chain data storage can be a private cloud on the client's infrastructure or a public storage provided by a third party or network. The flexibility of using cloud to store data depends on the implementation. Some peer-to-peer data storage are designed to be friendly to blockchain, such as IPFS[19] and Storj[20]. IPFS is free, but ensuring availability requires providing an IPFS server that hosts the data. The cost of Storj is US$0.015GB/month. In a peer-to-peer data storage, the data is replicated automatically by the peer-to-peer network, or based on the behaviour of users, *e.g.*, data is replicated once a user accesses it. In a cloud environment, data replication needs to be managed by the system or consumer.

2) Item collection: The concept of data item collection is common on blockchains, *e.g.*, when using blockchains as a registry. The total cost to a client application of using smart contracts on a public blockchain (even when running a local node) is likely to be less than running a whole private blockchain infrastructure as an index of data items. Using a separate chain may be more flexible because the blockchain can be configured to better support items of various sorts. However, interoperation among blockchains can be difficult. For off-chain options, the considerations and trade-offs for item data apply to item collections as well.

3) Computation: Computation in a blockchain-based system can be performed on-chain (*e.g.*, through smart contracts) or off-chain. Different blockchains offer different levels of expressiveness for on-chain computation. For example, Bitcoin only allows simple scripts and conditions that must be satisfied to transfer Bitcoin payments. Ethereum allows more general (Turing-complete) programs, and these programs can not only perform conditional payments but also make modifications to the working data in smart contract variables. There are other smart contract languages which are more expressive than Bitcoin's simple scripts, but which are purposefully not Turing-complete, in order to facilitate static analysis. An example is the Digital Asset Modelling Language (DAML)[21], which is designed to codify financial rights and obligations.

Smart contracts are not processed until their invoking transactions are included in a new block. Blocks impose an order on transactions, thus resolving non-determinism which might otherwise affect their execution results [17]. One benefit of using on-chain computation, rather than using blockchain as a data layer only, is the inherent interoperability among the systems built on the same blockchain network. Other benefits are the neutrality of the execution environment and immutability of the program code once deployed. This facilitates building trust in the shared code among untrusting parties.

C. Architectural Design Regarding Blockchain Configuration

In the following, we discuss various configuration options for blockchains. An overview is given in Table III.

When using a blockchain, one design decision is the *scope*, *i.e.* whether to use a public blockchain, consortium/community blockchain or private blockchain [6]. Most digital currencies use public blockchains, which can be accessed by anyone on the Internet. Using a public blockchain results in better information transparency and auditability, but sacrifices performance and has a different cost model. In a public blockchain, data privacy relies on encryption or cryptographic hashes. A consortium blockchain is used across multiple organisations. The consensus process in a consortium blockchain is controlled by pre-authorised nodes. The right to read the blockchain may be public or may be restricted to specific participants. In a private blockchain network, write permission is kept within one organisation, although this may include multiple divisions of a single organisation[22]. Whether using a consortium blockchain, private blockchain or permissioned public blockchain[23], a permission management component will be required to authorise participants within the network. Private blockchains are the most flexible for configuration because the network is governed and hosted by a single organisation. Many blockchain platforms support deployment as consortium blockchains or private blockchains, e.g., Multichain and Eris.

In Bitcoin, the blockchain *data structure* is a chain of blocks, and when conflicts occur the longest chain is selected by participants. This approach can been modified to improve scalability [24]. In the Greedy Heaviest-Observed Sub-Tree (GHOST) [19] protocol, miners reference competing independently-mined blocks (*uncle* blocks), to add weight to their chain for its selection as the main chain. This recognition of concurrent work allows shorter inter-block times which can improve throughput. Other proposed changes have included changing the chain from a list to a directed acyclic graph (DAG) [13] to allow non-conflicting transactions from uncle blocks to be incorporated in the main chain. Selection rules could be configured to select the longest chain or the heaviest sub-tree, by block length or by aggregate difficulty. The internal structure of blocks is also a design option. Segregated witness [27] has been proposed in the Bitcoin community to separate (segregate) signatures (witnesses) from the rest of the data in a transaction. The size of the witnesses then does not count towards the size limit within the block. In a blockchain network, all the transactions are replicated on every node, which increases overall storage requirements and can thus affect scalability. Pair-wise ledgers, like R3's Corda[24], have been proposed which would both improve privacy and scalability by reducing the replication of data across the network. However, this may negatively impact data integrity and availability.

[19] https://ipfs.io/

[20] https://storj.io/

[21] https://digitalasset.com/press/introducing-daml.html

[22] There is a grey area between consortium blockchains and private blockchains, and the differences may be more administrative than technical. Nonetheless we distinguish them here because at their extremes they have architectural differences.

[23] Ripple may be argued to be a permissioned private blockchain.

[24] https://github.com/corda/corda

Table III: Blockchain-related design decisions regarding blockchain configuration
(⊕: Less favourable, ⊕⊕: Neutral, ⊕⊕⊕: More favorable

Design Decision		**Option**	**Impact**			
			Fundamental properties	**Cost efficiency**	**Performance**	**Flexibility**
Blockchain scope		Public blockchain	⊕⊕⊕	⊕	⊕	⊕
		Consortium/community blockchain	⊕⊕	⊕⊕	⊕⊕	⊕⊕
		Private blockchain	⊕	⊕⊕⊕	⊕⊕⊕	⊕⊕⊕
Data structure		Blockchain	⊕⊕⊕	⊕	⊕	⊕
		GHOST	⊕⊕	⊕⊕	⊕⊕	⊕
		BlockDAG	⊕	⊕⊕⊕	⊕⊕⊕	⊕⊕⊕
		Segregated witness	⊕⊕⊕	⊕⊕	⊕	⊕
Consensus Protocol	Security-wise	Proof-of-work	⊕⊕⊕	⊕	⊕	⊕
		Proof-of-retrievability	⊕⊕⊕	⊕	⊕	⊕
		Proof-of-stake	⊕⊕	⊕⊕	⊕⊕	⊕⊕⊕
		BFT (Byzantine Fault Tolerance)	⊕	⊕⊕⊕	⊕⊕⊕	⊕
	Scalability-wise	Bitcoin-NG	⊕⊕⊕	⊕	⊕	⊕
		Off-chain transaction protocol	⊕	⊕⊕⊕	⊕⊕	⊕⊕⊕
		Mini-blockchain	⊕⊕	⊕⊕	⊕	⊕⊕
Protocol Configuration	Security-wise	X-block confirmation	⊕	⊕	⊕	⊕⊕⊕
		Checkpointing	⊕⊕⊕	⊕⊕⊕	⊕⊕⊕	⊕
	Scalability-wise	Original block size and frequency	⊕⊕⊕	n/a	⊕	n/a
		Increase block size / Decrease mining time	⊕	n/a	⊕⊕⊕	n/a
New blockchain	Security-wise	Merged mining	⊕⊕⊕	⊕⊕	⊕	⊕
		Hook popular blockchain at transaction level	⊕⊕	⊕	⊕⊕	⊕⊕⊕
		Proof-of-burn	⊕	⊕	⊕⊕⊕	⊕⊕
	Scalability-wise	Side-chains	⊕⊕⊕	⊕	⊕	⊕
		Multiple private blockchains	⊕	⊕⊕⊕	⊕⊕⊕	⊕⊕⊕

(Because they do not have a replicated global ledger, such systems are arguably not blockchains, but nonetheless still implement a kind of distributed ledger.)

The choice of *consensus protocol* impacts security and scalability. Once a new block is generated by a miner, the miner propagates the block to its connected peers in the blockchain network. However, miners may encounter different competing new blocks, and resolve this using the blockchain's consensus mechanisms. Usually the approach is fixed for a particular blockchain, but Hyperledger[25] is a framework with a modular architecture that caters for pluggable implementations of various consensus protocols.

The typical overall approach is called Nakamoto consensus [16], and relies on participants to select the longest chain of blocks they have observed at any point in time. In Bitcoin, new blocks are generated through a *proof-of-work* mechanism. This is a puzzle which is easy to verify, but which to solve is both difficult and takes effectively random time. Bitcoin miners compete to solve this puzzle for each block, using large amounts of computer power (and hence electricity) to increase their chances of winning the competition for each block. The investment required by miners for this acts to align their incentives with the good operation of the overall system. There are various proof-of-work mechanisms available from, *e.g.*, Ethash[26] used by Ethereum or Hashcash[27] used by Bitcoin. The mechanism in Primecoin[28] generates prime number chains which are also of interest to mathematical research. More recently, Permacoin proposes a modification to Bitcoin [15], which uses "proof-of-retrievability" to re-purpose Bitcoin's mining resource to distributed storage of archival data.

Proof-of-stake is an alternative mechanism for Nakamoto consensus, which selects the next mining node based on their holding of the native digital currency of the blockchain network. For example, the miners in Peercoin[29] need to prove the ownership of a certain amount of peercoin currency to mine blocks. Thus, proof-of-stake naturally aligns the incentives of digital currency holders in the blockchain with the good operation of the blockchain. There are various proof-of-stake protocols, *e.g.*, Tendermint[30] used in Eris, and Casper[31] used in Ethereum. These have different design goals, favouring one

[25] https://www.hyperledger.org/

[26] https://github.com/ethereum/wiki/wiki/Ethash

[27] https://en.bitcoin.it/wiki/Hashcash

[28] http://primecoin.io/

[29] https://peercoin.net/

[30] http://tendermint.com/

[31] https://github.com/ethereum/economic-modeling/tree/master/casper

non-functional property over another. Proof-of-stake does not necessarily select the next miner based on largest stakeholding, *e.g.*, Nxt[32] also uses a random factor, and Peercoin combines randomisation and coin age. BitShares[33] uses *delegated proof-of-stake*, where the accounts may delegate their stake to other accounts, rather than participating in the process of validating transactions directly. Compared with proof-of-work, proof-of-stake is more cost-efficient because much less computational power is used in mining, while latency is shorter [10]. However, passive holding of assets may become harder. A detailed comparison of different proof-of-work and proof-of-stake algorithms is out of the scope of this paper.

The *Byzantine fault tolerance (BFT)* protocol has been applied for consensus in permissioned blockchains, *e.g.*, in Stellar[34]. BFT ensures consensus despite arbitrary behaviour from some fraction of participants. Compared to Nakamoto consensus, it is a more conventional approach within distributed systems. Roughly speaking, BFT-based blockchains offers a much stronger consistency guarantee and lower latency, but for a smaller number of participants [24]. The core of Tendermint is also a BFT protocol, but uses a proof-of-stake mechanism to prevent Sybil attacks. BFT requires that all participants must agree on the list of participants in the network. Thus, the protocol is normally only used in permissioned blockchains.

Some new protocols have been proposed to improve *scalability*. Bitcoin-NG [8] decouples Bitcoin's blockchain operation into two planes: leader election and transaction serialisation. Once a leader is selected, it is entitled to serialise transactions until the next leader is selected. Thus, the leader election in Bitcoin-NG is forward-looking, and ensures that the system is able to continually process transactions. The Bitcoin lightning network [12] moves some of the transactions off-chain by establishing a multi-signature transaction between two participants as a micro-payment channel to transfer value off-chain. Once both sides wish to close the micro-payment channel and finalise the value transfer, a transaction is submitted to the global Bitcoin blockchain. Such bidirectional channels can be connected to establish a payment network leveraging Bitcoin. The intermediate transactions occurring in the payment channel are not included in the blockchain. Raiden[35] is a similar project on Ethereum. Mini-blockchain is a scheme proposed by Cryptonite[36]. The Cryptonite network maintains an account tree that holds the balance of all addresses, and a proof chain that stores all the historical block headers. The account tree is updated according to the transactions, and after a period of time, the transactions are forgotten by the network. Neither off-chain transactions nor mini-blockchain stores all the transactions on blockchain, thus, both of them sacrifice the fundamental properties of blockchain. Mini-blockchain saves space through forgetting historical transactions, but the performance is not necessarily better as the consensus mechanism is still the same.

Protocol configuration also affects security and scalability. As for *security*, a system using Nakamoto consensus has to deal with the possibility that the most recent few blocks get replaced by a competing fork. Different strategies have been used to confirm that a transaction is securely appended to the blockchain to prevent double spending. The first option is to wait for a certain number (X) of blocks to have been generated after the transaction is included into the blockchain. Commonly used values for X are 6 for Bitcoin, and 12 for Ethereum. The value 6 of Bitcoin blockchain was chosen based on the assumption that an attacker is unlikely to amass more than 10% of the hashrate, and that a negligible risk of less than 0.1% is acceptable[37]. The second option is to add a checkpoint to the blockchain, so that all the participants will accept the transactions up to the checkpoint as valid and irreversible. If a fork starts from a block before the checkpoint occurs, it is not accepted by any of the participants. Checkpointing relies on an entity trusted by the community to define the checkpoint, while X-block confirmation can be decided by the developers of the applications using blockchain.

Consensus protocols can be configured to improve *scalability* in terms of transaction processing rate. For example, there are some proposals for Bitcoin to increase its block size from 1MB to 8MB, to include more transactions into a block and thus increase maximum throughput. Another configuration change would be to adjust mining difficulty to shorten the time required to generate a block, thus reducing latency and increasing throughput. However, a shorter inter-block time would lead to an increased frequency of forks. Users would respond by waiting for more confirmation blocks [7].

When building a *new blockchain*, different strategies can be used to achieve security and scalability. For *security*, the new blockchain can be aligned with public blockchains, utilise exiting infrastructure, resources and trust. The first option is merged mining, which reuses the mining power of an existing public blockchain to mine and secure the new blockchain. In this case, a proof-of-work found by the miner of the public blockchain is used by two blockchains with possibly different difficulty levels. First, the miner produces a transaction set for both blockchains. The hash of the block produced for the new blockchain is added to the public blockchain. Then, once the miner finds a proof-of-work at the difficulty level of either of both blockchains, the proof-of-work is combined with the transaction set, and submitted to the corresponding blockchain. Namecoin is the first blockchain that uses merged mining with the Bitcoin blockchain. Merged mining reuses an established blockchain network, but it might be difficult initially to persuade the miners of the existing blockchain to join a new blockchain network. A more loosely coupled way is to hook into a public blockchain by adding hashes of the new blockchain to transactions of the public blockchain. For instance, Factom[38] anchors into Bitcoin blockchain every 10 minutes by submitting a transaction to the Bitcoin blockchain with a current hash of

[32] https://nxt.org/
[33] https://bitshares.org/
[34] https://www.stellar.org/
[35] https://github.com/raiden-network/raiden
[36] http://cryptonite.info/
[37] https://en.bitcoin.it/wiki/Confirmation
[38] http://factom.org/

the Factom blockchain. The third option is proof-of-burn. The purpose of proof-of-burn is to verifiably destroy tokens rather than generating tokens. To "transfer" tokens from a public blockchain to the new blockchain, the miners need to provide proof that their tokens were sent to a verifiably unspendable address. The burnt tokens, originally mined by proof-of-work, represent the corresponding computational power. Proof-of-burn can be used for bootstrapping a new cryptocurrency, *e.g.*, Counterparty[39], as it ensures serious commitment.

Rather than using a unique chain to record all types of transactions, multiple blockchains can be used to isolate information of separate concerns and with different characteristics, and to improve scalability. The first option is to use a side-chain [4]. Side-chaining is a mechanism that allow tokens of one blockchain to be securely transferred and used in another blockchain and still can be securely moved back to the original chain. The original chain is called *main chain*, and the one that accepts the tokens from the original chain is called *side-chain*. Multiple private chains could be used to separate concerns, where each of the private chains could link with a public blockchain. Side-chains can help to build a blockchain ecosystem based on a popular main blockchain, without significantly increasing the load on the main chain. However, the clients of side-chains may become complex, because they typically need to be able to process transactions from the main chain and the side-chain.

D. Other Architectural Designs and Deployment

Other design choices concern anonymity and incentives. Finally we discuss the impact of deployment.

1) Anonymity: Although the Bitcoin blockchain is perceived to be anonymous, research has shown that Bitcoin transactions can be linked to compromise the anonymity of Bitcoin users. Different techniques have been proposed to preserve anonymity on blockchain. Zcash[40], also called Zerocash or Zerocoin, encrypts the payment information in the transactions and uses a cryptographic method to verify the validity of the encrypted transactions. A zero-knowledge proof construction is used to allow the blockchain network to maintain a secure ledger and enable private payment without disclosing the parties or amounts involved. A *mixing* service groups several transactions together so that a payment contains multiple input addresses and multiple output addresses. Anonymity is preserved because it is hard to track which output address is paid by which input address. To further improve the way that mixing service operates, a series of mixing services can be linked sequentially. If mixing transactions are uniform in value, this minimises the trace between input and output addresses. A centralised mixing service requires a third party to operate, e.g., CoinJoin[41] and Blindcoin [23]. Distributed mixing services, on the other hand, do not rely on a single third-party, e.g., CoinSwap[42].

2) Incentive: Blockchains and their applications (especially on public blockchains) introduce financial incentives (or reputation and rating mechanisms) for miners to join the network, validate transactions and generate blocks correctly. For example, in Bitcoin, miners have two incentives: the reward for generating new blocks and the fees associated with transactions. Miners in Ethereum also charge a fee to execute smart contracts. Enigma[43] has a fixed price for storage, data retrieval, and computation within the network. Enigma also requires a security deposit for nodes to join the network. If a node is found to lie, its deposit will be split among the honest nodes. Reputation and rating mechanisms are generally used in peer-to-peer systems as a sign of trustworthiness of the peer as judged by others.

3) Deployment: Deployment of blockchain also has impact on the quality attributes of the system. For example, deploying a blockchain on a cloud provided by third-party, or using a blockchain-as-a-service model directly introduces the uncertainty of cloud infrastructure into the system. Here the cloud provider becomes a trusted third-party and a potential single point of failure for the system. Deploying a public blockchain system on a virtual private network can make it a private blockchain, with permissioned access controls provided at the network level. However the virtual private network will introduce its own additional latency overhead.

IV. Design Process for Blockchain-based Systems

Taxonomy can be used during the process of architecting software systems to guid the system design [11]. In this section, we propose an indicative conceptual model of architecting a system that potentially uses blockchain technology. The process, shown in Fig. 1, is used to illustrate how our taxonomy can be used to guide the system design at different stages of the design process. The process starts from the decision to decentralise trust (authority) – or not. A blockchain is used in scenarios where no single trusted authority is required and the trusted authority can be decentralised or partially decentralised. Design decisions regarding trust decentralisation are discussed in Table I. Given the limitations of blockchains, the next major decision is splitting computation and data storage between on-chain and off-chain components. The respective design decisions are discussed in Table II.

After that, a collection of design decisions around blockchain configuration need to be made, like the type of blockchain, consensus protocol, block size and frequency. The design decisions on blockchain configuration are discussed in Table III. Other design decisions are discussed in Section III-D. The arrows only illustrate one of the possible sequences to make design decisions. Some decisions mainly affect scalability (like block size and frequency), security (like consensus protocol), cost efficiency (like type of blockchain) and performance (like data structure). There are also trade-offs between the fundamental properties of blockchain. Finally, where to deploy the modules of the blockchain-based system is also important.

Every step in Fig. 1 is itself a procedure to decide between al-

[39] http://counterparty.io/

[40] https://z.cash/

[41] https://bitcointalk.org/index.php?topic=279249.0

[42] https://bitcointalk.org/index.php?topic=321228.0

[43] http://enigma.media.mit.edu/

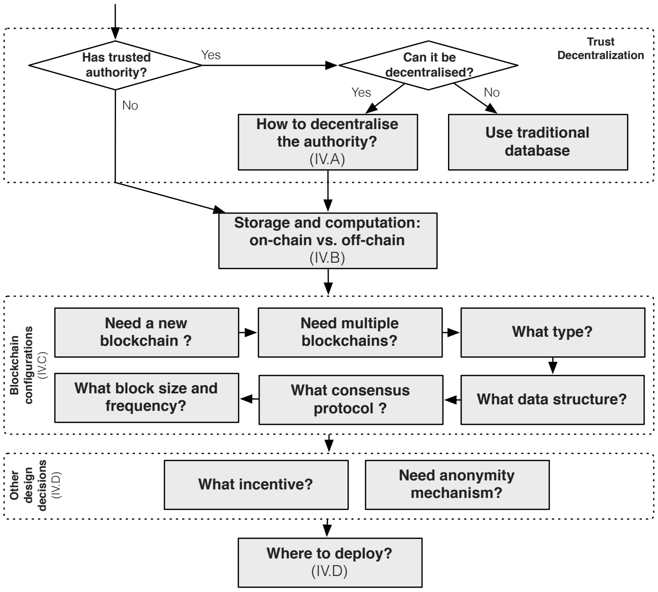

Figure 1: Design process for blockchain-based systems

ternative options. Throughout this design process, our taxonomy can assist the decision making through enabling a systematic comparison among the capabilities of different design options. The taxonomy also shows the impact of different design options on the quality attributes. The trade-off analysis of quality attributes provides a foundation for the comparison.

V. Conclusion

Blockchain is an emerging technology for decentralised and transactional data sharing across a large network of participant who do not need to trust each other. It enables new forms of distributed software architectures, where components can find agreement on their shared states without trusting a central integration point or any particular participating components. Blockchain, as a software connector with a complex internal structure, has various configurations and different variants. Using blockchain in different scenarios requires the comparison of blockchain options and products with different implementations and configurations.

In this paper, we propose a taxonomy of blockchains and blockchain-based systems. The taxonomy can be used when comparing blockchains and assist in the design and evaluation of software architectures using blockchain technology. Our taxonomy captures the major architectural characteristics of blockchains, and the impact of different decision decisions. This taxonomy is intended to help with important architectural considerations about the performance and quality attributes (*e.g.*, availability, security and performance) of blockchain-based systems. Other than taxonomy, patterns is also a mechanism to classify and organise the existing solutions.

In our future work, we plan to propose a list of design patterns for applications based on blockchain. We also plan to investigate the existing patterns for distributed system, peer-to-peer system and software in general and asses the applicability of the exisiting patterns to the blockchain-based applications.

References

[1] Distributed ledger technology: beyond blockchain. Technical report, 2016. UK Government Chief Scientific Adviser.

[2] The future of financial infrastructure - an ambitious look at how blockchain can reshape financial services. Technical report, 2016. WEF report.

[3] M. Ali, J. Nelson, R. Shea, and M. J. Freedman. Blockstack: A global naming and storage system secured by blockchains. In *USENIX ATC*, Santa Clara, CA, 2016.

[4] A. Back, G. Maxwell, M. Corallo, M. Friedenbach, and L. Dashjr. Enabling blockchain innovations with pegged sidechains. 2014.

[5] R. G. Brown. The "Unbundling of trust": How to identify good cryptocurrency opportunities?, 2014.

[6] V. Buterin. On public and private blockchains, 2015/08.

[7] C. Decker and R. Wattenhofer. Information propagation in the Bitcoin network. In *P2P*, Trento, Italy, 2013.

[8] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse. Bitcoin-NG: A scalable blockchain protocol. In *USENIX NSDI*, Santa Clara, CA, 2016.

[9] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *FC*, Christ Church, Barbados, 2014.

[10] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Čapkun. On the security and performance of proof of work blockchains. In *ACM CCS 2016)*, Vienna, Austria, 2016.

[11] I. Gorton, J. Klein, and A. Nurgaliev. Architecture knowledge for evaluating scalable databases. In *WICSA*, Montréal, Canada, 2015.

[12] T. D. Joseph Poon. The Bitcoin lightning network: Scalable off-chain instant payments. 2016.

[13] Y. Lewenberg, Y. Sompolinsky, and A. Zohar. Inclusive block chain protocols. In *FC*, San Juan, Puerto Rico, 2015.

[14] N. R. Mehta, N. Medvidovic, and S. Phadke. Towards a taxonomy of software connectors. In *ICSE*, 2000.

[15] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz. Permacoin: Repurposing Bitcoin work for data preservation. In *IEEE S&P*, San Jose, CA, 2014.

[16] S. Nakamoto. Bitcoin: A Peer-to-Peer electronic cash system, 2008.

[17] C. Natoli and V. Gramoli. The blockchain anomaly. In *NCA'16*. IEEE, Oct 2016.

[18] S. Omohundro. Cryptocurrencies, smart contracts, and artificial intelligence. *AI Matters*, 1(2):19–21, Dec. 2014.

[19] Y. Sompolinsky and A. Zohar. Accelerating Bitcoin's transaction processing - fast money grows on trees, not chains. Cryptology ePrint Archive, Report 2013/881, 2013.

[20] M. Swan. *Blockchain: Blueprint for a New Economy*. O'Reilly, US, 2015.

[21] T. Swanson. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. 2015.

[22] F. Tschorsch and B. Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):464, 2016.

[23] L. Valenta and B. Rowan. Blindcoin: Blinded, accountable mixes for Bitcoin. In *FC*, San Juan, Puerto Rico, 2015.

[24] M. Vukolić. The quest for scalable blockchain Fabric: Proof-of-Work vs. BFT replication. In *iNetSec*, Zurich, Switzerland, 2015.

[25] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling. Untrusted business process monitoring and execution using blockchain. In *BPM*, Rio de Janeiro, Brazil, Sept. 2016.

[26] G. Wood. Ethereum: A secure decentralized generalised transaction ledger — homestead draft. Technical report, 2016.

[27] P. Wuille. Segregated witness and deploying it for Bitcoin, 12 2015.

[28] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen. The blockchain as a software connector. In *WICSA*, 2016.