Notes on Blockchain Database Systems

Henry F. Korth

May 1, 2021

Abstract

Blockchain systems exhibit a variety of features and internal structure. As blockchain technology moves beyond the early (though still dominant) blockchains to applications in finance, national currencies, and enterprise information systems, it is useful to look at blockchain system structure from the standpoint of database systems. This paper seeks to describe blockchain structure in terms inspired by the traditional 3-tier database architecture and the system structure of full-featured database systems. Unlike many published blockchain white papers, we seek a fine-grained level-based division of data and of functionality in order to enable both a mixing of innovative concepts from existing systems, and the identification of focused subareas for future research. After addressing these issues in terms of data organization, validation, validator selection, and general governance, we consider the matters of privacy and external regulation; the former a traditional motivation for blockchains, and the latter an increasing focus of emerging government-overseen deployments. We conclude by applying our concepts to multi-blockchain systems and frameworks that enable them.

1 Introduction

Blockchain burst into wide awareness with the release of Bitcoin in 2009[27]. In a rapid series of new blockchain releases, blockchain systems evolved towards a richer model of data and computation. Ethereum[9] marked the transition of blockchain from a log of currency debit-credit transactions to a new type of "world-wide compute engine" that could serve as a platform for other systems (now known as *layer-2* systems). Hyperledger Fabric[3] is perhaps the leading example of the concepts of blockchain serving as a foundation for decentralized enterprise-class information systems. These three are just the most notable of many such systems and more continue to appear.

At the most fundamental level, a blockchain system stores *data* and supports transactions over those data, and, thus, can be considered a form of database system.[1] From that starting point, this paper attempts to describe blockchain systems using the terminology and conceptual foundations of database system research. There are several reasons for doing this:

- Blockchain system development is repeating some of the history of database development in a community where classic database concepts are less familiar. By casting blockchain system concepts in database terms, this paper seeks to point out areas where the database research community can impact existing technology trends in the blockchain community.

- Enterprises are looking increasingly at blockchain solutions for a variety of applications. Some, unfortunately, are drawn to blockchain by the hype and its perceived trendiness. This paper seeks to provide an abstraction of blockchain concepts to facilitate their deployment in any database context as opposed to suggesting deployment of a particular blockchain system that includes database-like features, such as BigchainDB[6].

- Many published blockchain system whitepapers fail to separate out the various design decisions made. We propose a taxonomy of blockchain system components along with a "layered" description of the

definition motivated by the physical / logical model framework for database systems. This framework facilitates choosing, from a list of features inspired by several existing systems, those suitable for a particular deployment. We put the term "layered" in quotes because our taxonomy is not truly a hierarchical layering since some systems use a more-general graph structure of our "layers". That said, most popular blockchain systems fit our layering in a hierarchical manner.

- Our proposed component taxonomy identifies focused areas for future research.

The motivation for this work comes from the way in which the layers and components of database systems have been clearly defined in a way that enables research and development efforts to address a single issue (e.g., query optimization), without having to describe an entirely new system.

We shall not provide an introduction to blockchain here. There are many such introductions in blockchain whitepapers and in a variety of texts, including [17, 28, 35]. We also assume familiarity with Byzantine fault tolerance[30], public-key encryption and digital signatures[11]).

2 High-Level Overview of Blockchain System Structure

Blockchain systems are often described in the literature in terms of a specific blockchain. As a result, the data in such a system are described in manner commingled with the description of a specific data structure. The updates to the data are most often described operationally. This is unfortunate, since the primary value of blockchain technology is not any particular data structure (it indeed does not have be a "chain of blocks.") but rather a set of *blockchain properties* applied to a distributed database. These properties are:

- **Decentralization:** Control of the database in all aspects is based on consensus of a large number of participating nodes, with provision not only for fault-tolerance but also malicious behavior by some minority of the nodes. The degree of decentralization varies among blockchain systems and need not be uniform across all levels and functions of the system. Decentralization is an important property of most public blockchains (e..g., Bitcoin), but less so in enterprise-class, permissioned blockchains such as Hyperledger Fabric.

- **Tamper resistance:** Any effort by a participating node to alter committed data is detectable and actionable via the consensus mechanism.

- **Irrefutability:** Transactions, once submitted, are permanently traceable to their submitter (or participants for multi-signature transactions) via digital signatures.

- **Anonymity:** Users of the system may keep their blockchain identities separate from their real-world identities. The degree of anonymity varies widely from strong anonymity supported by zero-knowledge techniques[7] to anonymity only to certain users but not to a controlling authority (as for certain enterprise blockchains and certain central-bank digital currency proposals). Bitcoin lies between these extremes.

Arguably, if implementors of an application have little or no interest in these properties, a traditional database is a highly preferable deployment strategy.

A specific system may implement the above properties using a variety of data structures and algorithms, the choice of which should ideally be separate from a definition of the desired blockchain properties.

Similarly, blockchain transactions are often described operationally, rather than in the ACID framework familiar to database researchers and practitioners. Indeed blockchain transactions do not enjoy the full set of ACID properties. Some of the distinctions between database transactions and blockchain transactions are immediately obvious (e.g., the limited nature of a Bitcoin transaction), but others are subtle with possibly unintended consequences, such as the ability of one transaction to be designed based on the code of another transaction due to the publicly readable status of the pool of pending transactions. The interaction of blockchain transactions can depend on whether the blockchain equivalent of the database transaction

manager (i.e., the miner) takes an active role in choosing an ordering of transactions, possibly to its own advantage. Such behavior is referred to as *front-running*.

In this paper, we attempt to structure a blockchain system in componentized and largely, though not completely, layered framework. This componentization enables us to show how various functions within a blockchain system control or influence behavior and to provide a framework for describing precisely what that influence is. Our focus here is descriptive. We see this a basis for future work in a variety of areas including:

- Design of blockchains targeted to a specific application.
- Component reuse.
- Design of effective and efficient control and oversight mechanisms in a blockchain subject to government regulation (as in a central-bank digital currency or a blockchain seeking approval from a regulatory authority).
- Design of regulated systems in a way that provides an easily understood degree of transparency and perhaps limitations on central control and oversight.

The structural approach we take here is intentionally agnostic of the actual specific purpose of the blockchain system. From the perspective of a database researcher, than may make sense but for a blockchain system, the exclusion of purpose ignores one of the greatest motivators for public blockchains: privacy and anonymity. The actual implementation of blockchain objectives as regards privacy and anonymity can be described using our level-based approach. Later in this paper, in Section 10, we address how specific policies relate to our organizational taxonomy.

3 Physical Level

Although the term *blockchain* arises from the linked list of blocks that forms the underlying data structure of the Bitcoin blockchain, it is certainly possible to use any physical-level implementation to provide the same functionality at the logical level. Several existing blockchain systems use a physical model substantially different from Bitcoin, such as the directed-acyclic-graph model of Iota[32] (in which the DAG is called a *tangle*). Hyperledger Fabric stores blockchain data in a NoSQL database. Bitcoin has used (in various versions) both Berkeley DB and LevelDB to store index data. Ethereum clients use RockDB and LevelDB among others.

The goal of a physical model for a blockchain is to represent the fundamental data elements of the blockchain along with (perhaps) some data structures to facilitate standard accesses and "updates". We quote the term *update* here because blockchain immutability implies append-only operation, while in fact the operations performed may include the semantic equivalent of an update, as is the case, for example in an Ethereum funds-transfer operation. The exact set of fundamental data items varies by blockchain but typically includes some representation of transactions, account balances, unspent transactions, pending transactions, code plus associated variables or stacks, among other items.

In the typical three-tier database architecture, the physical model needs only to present an interface to higher levels. Blockchain systems, due to their inherent distributed nature, may face an additional physical-level compatibility requirement across different architectures and code bases. Ethereum, for example, offers a set of clients from which the operator of a node may choose. The existence of multiple clients provides the blockchain network with some degree of robustness to bugs in the client software. But that comes with the requirement that the underlying data structures must have a degree of language independence.

The distributed nature of a blockchain system plus its (usually) decentralized control requires a degree of software-upgrade compatibility at the physical level. In centrally managed systems, an upgrade is easier to manage than in a decentralized system. Older and newer implementations may transfer blockchain data among themselves. If that is not possible, then the upgrade would constitute a hard fork, a much more

difficult form of upgrade to accomplish at the blockchain governance level, which we discuss in Section 9. Existing blockchains have forked many times for physical-level implementation changes (such as block size or off-chain storage of certain data) and disputes over these matters have led to ongoing forks of various blockchains into separate independent chains. A better-defined physical-level interface could perhaps have avoided some of these issues, though not, of course those that resulted from governance issues.

As a result, it may not be ideal to view the physical level as a single, independent component. Many blockchain systems include in their physical model certain index structures that do not need cross-node data-structure compatibility. The issues of physical-level compatibility can be reduced by separating out those aspects of the physical model that represent data that are transferred directly among independent nodes and those aspects that serve only the needs of local performance optimization. Expanding on this principle, we may seek to define the physical model in a manner that permits data items to be converted simply and rapidly from one physical representation to another. This would allow the vertical, three-tier-architecture concept of physical data independence to be expanded to a horizontal decentralized concept of physical data independence.

Further subtleties at the physical layer derive from the distinct requirements of light clients (as opposed to full nodes). Light clients store much less data than full nodes and depend on full nodes for their remaining data needs. Physical-layer compatibility is required for certain exchanged data structures (e.g., for Merkle proofs, as we discuss next).

Up to this point, we have viewed the physical level primarily as an encoding of data describing blockchain activity. Blockchain systems, unlike database systems, need to store additional data to secure the blockchain from attack and to enable proofs of correctness of data returned. The exact requirements in this regard depend on the level of trust among nodes in the blockchain system. In a database system, where trust is absolute, no such data are needed at all. Some aspects of these security-mandated data can be abstracted at the logical level (for example, the hash of a block), but others can be viewed as the domain of the physical level (for example, the specific structures used to implement a Merkle tree are the domain of the physical level). The interface between physical and logical models for Merkle trees[24] and Merkle/Patricia trees[26] is interesting. The logical level needs to model a description of Merkle proofs and data updates (such as balance transfers in Ethereum) even though the actual implementation of those items at the physical level should be independent of the logical modeling of them.

Another aspect of physical-level modeling is the location of storage of blockchain data. Up to this point, we have treated all blockchain data as if it were stored on-chain at all nodes. There are other physical storage models that present an equivalent logical-level view. Significant volumes of blockchain data can be stored off-chain secured by a Merkle root stored on-chain. Notable instances of this approach include Ethereum and post-SegWit Bitcoin.

4 Logical Level

The logical level of a database describes what is stored at the physical level and how those items relate to each other. The logical-level abstractions in the database world omit implementation details to a greater or lesser degree depending on the model being used (more appear in the now-deprecated CODASYL DBTG model than in the now-ubiquitous relational model). As is the case with the multiple logical models in the database world, one can define multiple models for blockchain systems. This definitional problem is part art and part science, since one should seek to define models broadly enough to encompass a wide variety of existing blockchain systems.

But here too at logical level, certain properties of blockchains create interesting conceptual distinctions from database logical modeling. Many blockchains store what amounts, in database terminology, to a simple log. That is the case for Bitcoin, where, if one ignores the hashes stored for chain security, the data is simply a list of transactions. That list is structured very differently from a database log (where transaction steps are interleaved) but it is, at the highest level, a list of transactions, which appears easy to model at the logical level. More troublesome however, is the question of what is really "in" the log. In a database

system, log records stored only in volatile storage are not permanent. They are used in a running system to abort a transaction, but disappear in a crash. A log record becomes permanent when it is written to nonvolatile storage with whatever degree of redundancy is deemed necessary for permanence. There is a similar notion in blockchain centered around the concept of *finality*. While some systems add a transaction to the blockchain only when it is truly committed, most add transactions that are only likely to become final, or permanent. That degree of likelihood increases the longer the transaction remains part of the blockchain until the probability of its removal becomes negligibly small, leading to the transaction being deemed final. Non-final transactions need to be visible since that is how the blockchain grows and leads to non-final transactions becoming final. This results in a shared log among decentralized systems that lacks the rigorous append-only structure of a database system. While the details of how this transaction log is implemented is a physical-level issue, the semantics of this log are of critical importance at the logical level, since those semantics influence the various applications that use the blockchain. Those semantics are also a key part of the operation of the consensus (Section 8) and governance (Section 9) levels.

A well-designed logical level can provide a clean separation between the finality of blockchain data and how that finality is determined. Those concepts are mixed in many current blockchain systems, such as Bitcoin, where finality is defined based on the concept of the longest fork, even though the definition of longest fork is a node-dependent concept whose global uniformity is only a probabilistic concept. Database log permanence is based on a well-defined concept (typically the write-ahead log protocol), while the corresponding blockchain concept is less deterministic.

Logical modeling can simplify the definition of cross-chain transaction semantics if some set of standardized, well-defined concepts take hold across many blockchain systems. Current approaches depend directly on blockchain-specific features such as the type of time-lock features or smart contracts available. Standardization may be dismissed as mere wishful thinking, but newer blockchain implementations such as Ethereum 2 and Polkadot[8] combine aspects of cross-chain semantic issues within the more unified framework of a single blockchain.

5 Higher-Level Modeling

In a database system, the three-tier architecture includes as its top level, the user-specific *view level*. Other aspects of system operation such as the commit protocol in a centrally administered distributed database are system-internal issues unrelated to data semantics. This is not the case for a blockchain system. The manner in which distributed consensus is obtained and who participated in that consensus (and in what role) are often important data items in the blockchain that are used for validation of the data. The types of data involved in consensus and validation vary among blockchains. Those data can be part of the underlying logical data model, but the algorithms used for consensus and validation themselves can be componentized and abstracted. Blockchains might share the same concept of a valid block but differ on the means of selecting a particular valid block. This leads us to separate the adding of blocks to a blockchain into multiple levels of modeling:

1. **Validation level:** Section 6 discusses the description of a valid block in terms of a logical-level model, leaving the choice of a specific implementation of validation, including any performance-improving data structures, to either lower levels of modeling or to system-implementation details.

   One aspect of block validity may be whether the block proposer is indeed allowed to make such a proposal. This would require identifying the proposer along with proof that the proposer was selected correctly. Some blockchain systems perform a separate consensus regarding the designated proposer of the next block and the decision as to whether that proposed block in valid. That separate consensus takes place using techniques described in the consensus level.

2. **Validator selection level:** Section 7 discusses the description of a valid set of validator nodes. Often this is simple, as in a Bitcoin-style proof-of-work system in which every full node is a validator. However, some systems' key competitive advantage may lie at this level, as illustrated for example by

Algorand's[16] *cryptographic sortition* approach. The validator selection process itself may take the form of consensus protocol. It is possible for consensus on a set of validators to use a protocol distinct from the protocol used to reach consensus on the next block.

3. **Consensus level:** Section 8 discusses the protocols by which the selected set of validators reach consensus on whatever it is on which they seek to agree. Typically this is the next block to be added to the blockchain, but it may be some other aspect of blockchain operation, such as the set of block validators, expulsion or slashing of misbehaving nodes, etc.

   These protocols are widely varied, including fully trustful consensus (such as Raft[29] for some Hyperledger Fabric deployments), variations of Byzantine consensus that permit a variety of trust models, variations of proof-of-stake and proof-of-work, and hybridizations of these.

Most of our higher levels specify a computation rather than just data. For example, a consensus protocol is a description of actions taken to reach consensus. Academic consensus protocols often provide a highly abstracted definition of what is to be agreed upon and focus on the protocol. In most cases of blockchain systems, the nature of what is to be agreed upon in a nontrivial aspect of consensus. This applies also to governance and validator selection. In each case, we seek to identify what can be described declaratively so that, at least in principle, one protocol could be replaced with another while not impacting the overall framework of the system.

6 Validation Level

The validation level of a blockchain system does not have a direct analog in the database three-tier architecture. It is a description, using the elements of the logical level description of the system, of what constitutes a valid blockchain. That might be considered a concept analogous to database consistency, except that for the most part, work on database transaction processing ignores the precise definition of consistency and instead just assumes that, whatever consistency may mean, transactions preserve it. Thus, a database transaction may be modeled as a mapping from the set of all consistent database states to the set of all consistent database states. The database-style approach to consistency does not work in a blockchain setting because it is not assumed that transactions are correct programs that preserve consistency. That is most certainly true in a public blockchain like Ethereum or Bitcoin,[1] but may also be the case in the most trustful permissioned blockchain settings. In even those latter high-trust environments, there is some degree of validation applied to transactions. For those reasons, a blockchain system needs a precise definition of transaction correctness. Furthermore, when transactions are grouped into blocks as is done by most, though not all systems (one exception is Iota[32]), there is a need for a precise definition of a correct block.

The goal at the validation level is a declarative specification of correctness. Often the correctness criteria reference substantial parts of the blockchain, making efficient optimization of validation an important consideration. Optimization, however, is not part of the specification and different validators are free to choose their own methods of validation.

In any blockchain system, there are syntactic requirements regarding the format of a transaction and the structure of a block plus semantic specifications, which are more complex. A few examples follow.

- The Bitcoin model of a transaction spending the output of previously unspent transactions requires enforcement relative to the entire history of the blockchain. If a block contains a pair of transactions that both spend the same transaction's output, then the ordering of those transactions determines which is valid in the block and which is not. Pending transactions are retained for eventual addition to the blockchain (the *mempool*) and so the consistency of that collection relative to the blockchain itself must be maintained. A valid Bitcoin block must hash to a value less that the designated target value as it existed in the blockchain system at the time the block was added.

[1] For example, in Bitcoin, it is not the transaction that performs a test that its inputs have not been previously spent.

These issues are syntactic in nature even though they require good data structures for them to be managed efficiently.

- Beyond the relatively simple issue of double-spending, there is the issue of front-running[13, 19, 33]. A miner node may reorder transactions from the mempool to its own advantage, perhaps inserting a new transaction of its own ahead of and/or after some other transaction. The result in this case may be a syntactically correct block that is nevertheless inappropriate in a semantic sense. Protection against front-running is far from ubiquitous, but one can imagine that future regulated blockchains (e.g., those sponsored by government agencies) may have some validity rules to eliminate or reduce front-running.

- In blockchain systems in which validators are chosen by stake or by a random process, the block may contain a cryptographic proof that the validator set was chosen correctly. This specification thus relates to the validator selection level, but focuses only on the description of the selection process and not the process itself.

7 Validator Selection Level

Here, our focus is on who or what does validation and not on how validation is done, nor what it means for a block to be valid. In some blockchain systems, the specification of validator selection is declarative and trivial: all nodes are validators. Other systems have more complex selection mechanisms. For those latter cases, some use deterministic algorithms hard-coded into the system, while others use a consensus process so that all nodes agree on the subset chosen to be validators. Validator selection may include selection of a leader, as is the case is many Byzantine-consensus protocols or it may treat all validators as peers, in a leaderless model[5].

The case of consensus-driven validator selection is interesting in its circularity. Consensus must be achieved over a set validators but determining that set of validators requires consensus. Obviously, any real blockchain system must be based on a cycle-breaking specification at some point in the process and most do so in a direct, simple manner. However, a clear separation of the objective of validator selection and the protocol for achieving consensus makes it possible to envision a rich variety of mechanisms with potentially interesting features. We present a few such possibilities at a high level below:

- The consensus/validation cycle could be a recursive process that reduces to a base case. For a permissioned blockchain, that base case could be $n = 1$, that is, the owner of a permissioned system. Alternatively, the base could be a randomized process, such as Algorand's use of verifiable random functions[16, 25]

- Certain earlier parts of the consensus/validation cycle could be "stickier" than others and persist over several blocks or until some node or nodes object.

- Validator set selection can be an implicitly determined side effect of a consensus algorithm (e.g., Byzantine consensus) in which malicious or malfunctioning nodes are excluded.

8 Consensus Level

The goal of blockchain specification at the consensus level is to focus on the process of agreement. The actual object of agreement should be an item specified at the logical or validation level.

We illustrate this separation in restating Bitcoin consensus. It is a simple, syntactic statement: the blockchain is the longest fully valid fork in a node's local copy of the blockchain, maintained by a gossip protocol with peer nodes, with length ties broken in an arbitrary manner. There is no consensus protocol beyond simple, timely gossiping. All of the issues pertaining to nonces, mining, double-spending, and so forth, are described at the validation level.

For many other blockchain systems, the specification of the consensus protocol is complex, perhaps a full paper in its own right (e.g., Stellar[23, 22]). A well-described consensus protocol leaves that upon which agreement is reached as a parameter to the consensus process. Similarly, such a specification should leave the set of nodes participating in consensus as a parameter to the consensus process, while the consensus process focuses on how those nodes behave. Academic papers typically meet these criteria, but blockchain white papers often merge these issues. One frequent case of merger of these issues has to do with the response to faulty or malicious nodes (as opposed to simply detecting and identifying them). We treat such responses as a governance issue. Consensus creates some degree of global knowledge about nodes, and that informs actions at the governance level. Thus actions such as exclusion from future consensus activity, slashing, etc., are not part of the consensus-level process. A survey of consensus approaches is given in [20].

Consensus protocols necessarily rest upon assumptions[15]. At the consensus level, those assumptions are exactly that, and are not questioned, nor is one concerned about maintaining the validity of those assumptions. Governance must ensure that assumptions made at the consensus level are valid (or at least ensured with a strong level of guaranteed).

9 Governance Level

The governance level defines the true nature of a blockchain system. Whether it is a public chain, a tightly controlled private chain, or some partially decentralized permissioned chain all are determined from the governance model for the blockchain. Governance is the most controversial aspect of blockchain systems. It defines how the conflicting needs of privacy, transparency, regulation, and decentralization may be combined or traded off. Governance addresses a variety of top-level blockchain-management issues including:

- Authorization and access
  - Physical membership of devices or networks in the blockchain system
  - Authorized users of the blockchain system
    - Admission: both who/what and when
    - Removal: both who/what and why
- Rewards and/or punishment of specific users (e.g. airdrops, slashing, and removal).
- Setting system-wide parameters for lower levels
  - block rewards
  - validity parameters (such as the hash target for Bitcoin block validity)

Aspects of blockchain governance may be decided by a consensus mechanism among stakeholders. That mechanism can be any consensus mechanism we discussed in Section 8. That consensus mechanism is, in turn, subject to some sort of governance, creating a cyclic posing of the question "Who governs the governors?". Clearly, any actual system must have a non-recursive base case.

Other aspects of governance may be decided by a human process. Permissioned blockchain systems may be governed by one or a group of business enterprises via an off-chain process. The power of the governing entity may be limited in code or via a traditional business agreement. Government-run blockchains (especially the rapidly emerging set of central-bank digital currencies (CBDC) [14, 18]) or tightly-regulated blockchains may take this form, with the government controlling some or all aspects of the highest levels of governance. Indeed, one reason governments may pursue blockchain-based technology is to achieve greater, not lesser, central control of governance.

Certain cryptocurrencies have an ownership distribution that gives one or a small group of owners outsized control over the governance process. This has often been the case with newly emerging blockchains in which closely-held governance serves as means of securing the chain against attack. Generally, such a governance structure includes a plan for eventual decentralization.

Governance also includes policy on trusted oracles and any processes for the validation of oracles.

10 Privacy, Anonymity, and Identity Management

As we noted in Section 2, the levels we present in our taxonomy avoid reference to any specific purpose of the blockchain system (beyond examples). That is appropriate for any general-purpose information system conceptual design, but there is one category of "purpose" that is so fundamental to the motivation for blockchains that is deserves attention: privacy and anonymity. Blockchains vary widely in their approach to these properties. Permissioned blockchains may be secured against data access by non-members. Public blockchains may publish the entire blockchain openly, but keep the mapping between user-ids and off-chain entities (e.g., people) secret to all or secret to only certain governors. Related, but separate from our concern here, is data encryption. On-chain data is mostly public and unencrypted, but certain off-chain data may be encrypted, as in, for example, storage services such as Filecoin (https://fil.org/).

The privacy objectives themselves may be best defined in terms of our levels. As an example, it may be considered highly important to retain the privacy of validators than that of general users in order to defend against denial-of-service attacks against the validation process. (One such example is Algorand[16].)

Identity management is the dual of privacy. The secure connection between a blockchain identity and a real-world identity is essential in a variety of applications, including the maintenance of public records (real-estate ownership, motor-vehicle records, voting rolls, etc.). Regulated financial systems need to establish this connection for *know-your customer* (KYC) compliance. This connection between on-chain and off-chain identity may be public or may be private to certain levels of governance.

Whatever the objectives of a specific chain as regards issues of privacy, the question of how, and at what level, those objectives are implemented is important. Insecurity at one level may impact guarantees made (incorrectly) at another. Alternative implementations may be acceptable if they provide the same external level of protection.

We see this separation of function as particularly useful as new techniques, largely based on zero-knowledge, are developed and deployed, whether for strong privacy guarantees or to enable selective disclosure of data only to auditors or regulatory authorities. Other zero-knowledge-based disclosures in a decentralized enterprise blockchain system may depend on a consensus process (8).

11 Extension to Multi-Blockchain Systems

There are a vast number of cross-chain transaction systems. Lightning's[31] framework for off-chain Bitcoin transactions was one of the first. Privacy in such cross-chain transactions is challenging to achieve. Levels of privacy in cross-chain transactions, including the possibility of limited disclosure is discussed in [10]. At present, a vast number of blockchains operate at what is called *layer 2*, running on top of an existing blockchain. Most of these are ERC-20 tokens implemented on top of Ethereum. Most recently, there is substantial interest in peer chains as exemplified by Polkadot[8, 36] or cooperating parallel chains as in the design of Ethereum 2 shards. Sharded blockchains present challenging synchronization issues, one approach to which appears in [2].

The Polkadot model is of particular interest since it has a single *relay chain* for overall coordination of a collection of independent *parachains*. There are specific requirements regarding how a parachain interfaces with the relay chain. The relay chain rewards nodes for providing security checks on parachains. The internal structure of parachains is otherwise unconstrained. Indeed, a parachain does not even have to be a "blockchain" in any specific sense as long as it provides the required interfaces. The relay chain can coordinate message passing among parachains, enabling the deployment of a set of parachains to construct a multi-parachain system using their own means of collaboration and/or coordination. This separation of function in Polkadot's design is very much in the same spirit as the level-based concepts proposed in this paper and we speculate that a careful separation of function like the one we propose can be a valuable tool in the design of multi-parachain systems as the Polkadot ecosystem develops.

12 Related Work

The connection between blockchain systems and database systems has been explored from a variety of angles, such as [4, 34]. We have noted earlier than certain blockchain data may be stored in a database. In many deployments of permissioned enterprise blockchains, virtually the entire data storage may be database-based. Performance analysis of blockchain systems is best done by using database-style benchmarking as was done by [12]. A more enterprise-centric framework for blockchain implementation is presented in [21].

As blockchain systems see use for increasingly sophisticated finance operations, the power of smart contracts is both an opportunity and a threat. Front-running, flash loans, and other techniques represent either an attempt at fraud or useful market maintenance depending on whose point of view one accepts. It is reasonable to expect that, in time, mechanisms will be needed to manage trading systems for reasons of investor protection, regulation, and taxation. Doing this in an effective manner may involve checks, rules, or review at various levels of the system. The current literature and informal presentations (many in blogs or on *Medium*) highlight the problems in this space. Comprehensive solutions that go beyond ad-hoc modifications of specific systems arguably depends on a more structured database-style view of blockchain systems.

13 Conclusions

Blockchain system design is only in its beginning. Bitcoin and Ethereum are not the final word; indeed Ethereum is in the midst of a major design transition. New architectures such as that of Polkadot and Algorand are more evidence that we are in a time of major change and revisitation of system design.

This framework can be used as a structured way to describe existing blockchain systems. As such, it can be seen as a pedagogical tool. However, a more valuable use of this framework is to facilitate the definition of blockchain systems that are a hybrid collection of "best of breed" solutions to various components of blockchain systems and also to facilitate focused research on singular aspects of blockchain systems that can then be plugged in to existing code bases.

An important near-term application of a structured approach to blockchain systems is the design of a central-bank digital currency (CBDC) in a manner consistent with the policy objectives of various nations and their central banks. See [14] for one such example. Such systems need not only to address the correctness and governance issues covered here, but also the need for performance sufficiently high for global-scale commerce. As noted in Section 11, our framework may help in the exploitation of parallel and special-purpose hardware to achieve their performance goals.

References

[1] D. Agrawal and A. ElAbbadi. Blockchains and databases: Opportunities and challenges for the permissioned and the permissionless. In J. Darmont, B. Novikov, and R. Wrembel, editors, *Advances in Databases and Information Systems. ADBIS 2020*, Lecture Notes in Computer Science, vol 12245. Springer, Cham, 2020.

[2] M. J. Amiri, D. Agrawal, and A. ElAbbadi. SharPer: sharding permissioned blockchains over network clusters. Cryptology ePrint Archive, Report 1910.00765, 2020.

[3] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick. Hyperledger Fabric: A distributed operating system for permissioned blockchains. In *Proc. Eurosys'18*, April 2018.

[4] D. T. T. Anh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang. Untangling blockchain: A data processing view of blockchain systems. *IEEE Transactions on Knowledge and Data Engineering*, 30(7):1366–1385, July 2018.

[5] K. Antoniadis, A. Desjardins, V. Gramoli, R. Guerraoui, and M. I. Zablotchi. Leaderless consensus. Technical report, EPFL, 2021.

[6] BigchainDB. BigchainDB 2.0: The blockchain database. Web document, 2018. https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf.

[7] D. Boneh and V. Shoup. *A Graduate Course in Applied Cryptography*. cryptobook.us, 2020.

[8] J. Burdges, A. Cevallos, P. Czaban, R. Habermeier, S. Hosseini, F. Lama, H. K. Alper, X. Luo, F. Shirazi, A. Stewart, and G. Wood. Overview of Polkadot and its design considerations. arXiv 2005.13456, 2020.

[9] V. Buterin. Ethereum: A next-generation smart contract and decentralized application platform. Web document., 2013. https://ethereum.org/en/whitepaper/.

[10] A. Deshpande and M. Herlihy. Privacy-preserving cross-chain atomic swaps. In M. Bernhard and others., editors, *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, vol 12063. Springer, Cham, 2020.

[11] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6), 1976.

[12] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan. Blockbench: A framework for analyzing private blockchains. In *Proc. ACM SIGMOD Conference on the Management of Data*, pages 1085–1100, 2017.

[13] S. Eskandari, S. Moosavi, and J. Clark. SoK: Transparent dishonesty: Front-running attacks on blockchain. Cryptology ePrint Archive, Report 1902.05164, 2019.

[14] Y. J. Fanusie and E. Jin. China's digital currency. Technical report, Center for a New American Security, 2021.

[15] M. J. Fischer, N. A. Lynch, and M. S. Patterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, October 1985.

[16] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine agreements for cryptocurrencies. In *Proc. 26th ACM Symposium on Operating Systems Principles*, pages 51–68, March 2017.

[17] S. Gupta, J. Hellings, and M. Sadoghi. *Fault Tolerant Distributed Transactions on Blockchain*. Morgan & Claybool Publishers, 2021.

[18] K. W. Hanley and H. F. Korth. US central bank digital currency governance policy for effective regulation, citizen privacy, and operational transparency. in preparation, 2021.

[19] R. Khalil, A. Gervais, and G. Felley. TEX - a securely scalable trustless exchange. Cryptology ePrint Archive, Report 2019/265, 2019. https://eprint.iacr.org/2019/265.

[20] H. F. Korth. Consensus in enterprise and financial blockchains: Assumptions and challenges. In *Proc. First Workshop on Blockchain and Distributed Ledger*, 2019.

[21] O. Labazova. Towards a framework for evaluation of blockchain implementations. In *Fortieth International Conference on Information Systems*. Association for Information Systems, 2019.

[22] M. Lokhava, G. Losa, D. Mazières, G. Hoare, N. Barry, E. Gafni, J. Jove, R. Malinowsky, and J. McCaleb. Stellar consensus by instantiation. In *Proc. 33rd International Symposium on Distributed Computing (DISC 2019)*, pages 80–96, 2019.

[23] G. Losa, E. Gafni, and D. Mazières. Fast and secure global payments with Stellar. In *Proc. 27th ACM Symposium on Operating Systems Principles*, 2019.

[24] R. C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology — CRYPTO '87*, pages 369–378. Springer Berlin Heidelberg, 1988.

[25] S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *Proc. 40th IEEE Symposium on Foundations of Computer Science*, 1979.

[26] D. R. Morrison. PATRICIA—Practical Algorithm To Retrieve Information Coded in Alphanumeric. *J. ACM*, 15(4):514–534, October 1968.

[27] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Web document., 2008. https://bitcoin.org/bitcoin.pdf.

[28] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[29] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. In *Proc. USENIX*

Annual Technical Conference, pages 305–320, 2014.

[30] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, April 1980.

[31] J. Poon and T. Dryja. The Bitcoin Lightning Network, scalable off-chain instant payments. Web document., 2016. `https://lightning.network/lightning-network-paper.pdf`.

[32] S. Popov. The tangle. Technical report, The Iota Foundation, 2018.

[33] K. Qin, L. Zhou, B. Livshits, and A. Gervais. Attacking the DeFi ecosystem with flash loans for fun and profit. arXiv 2003.03810, 2021.

[34] A. Sharma, F. M. Schuhknecht, D. Agrawal, and J. Dittrich. How to databasify a blockchain: the case of Hyperledger Fabric. Technical report, Universität Saarland, 2018.

[35] A. Silberschatz, H. F. Korth, and S. Sudarshan. *Database System Concepts, 7th edition*, chapter 26: Blockchain Databases. McGraw Hill Education, New York, NY, 2020.

[36] A. Stewart and E. Kokoris-Kogia. GRANDPA: A byzantine finality gadget. arXiv 2007.01560, 2020.