Introduction to Blockchain Technologies

ICANN Office of the Chief Technology Officer

Paul Hoffman
OCTO-040
17 October 2024

TABLE OF CONTENTS

1 INTRODUCTION 4

2 BASIC BLOCKCHAIN COMPONENTS 4

2.1 Transaction List, Rules, and Database 5
2.2 Coins 5
2.3 Accounts 6

3 THE LEDGER AT THE CORE OF BLOCKCHAINS 6

3.1 Blockchain Ledger Example 7
3.2 Publishing the Ledger 7

4 GETTING FROM TRANSACTIONS TO DATA 8

4.1 Transactions 8
4.2 Processing Rules and Programs 9
4.3 The Database Derived from the Ledger and Rules 9

5 PUTTING TRANSACTIONS INTO A BLOCKCHAIN 10

5.1 Bad Blocks and Bad Transactions 11
5.2 The Committer Model 11
5.3 The Proposer Model 12

6 TRUST MODELS FOR BLOCKCHAINS 13

6.1 Trusting the Origin Block 14
6.2 Trust Model for the Committer Model 14
6.3 Trust Model for the Proposer Model 14

7 CRYPTOGRAPHY USED IN BLOCKCHAINS 15

7.1 Hash Algorithms 15
7.2 Digital Signatures 15

8 ASSOCIATIONS BETWEEN BLOCKCHAINS 16

9 BLOCKCHAINS AND FINANCE 17

9.1 Coins 17
9.2 Additional Assets 17

10 CONCLUSION 17

This document is part of ICANN's Office of the Chief Technical Officer (OCTO) document series. Please see the OCTO publication page for a list of documents in the series. If you have questions or suggestions on any of these documents, please send them to octo@icann.org.

This document supports ICANN's strategic goal to improve the assessment of, and responsiveness to, new technologies that impact the security, stability, and resiliency of the Internet's unique identifier systems through greater engagement with relevant parties. It is part of ICANN's strategic objective to evolve the unique identifier systems in coordination and collaboration with relevant parties to continue to serve the needs of the global Internet user base.

1 Introduction

This document explains the technologies used in blockchains. It is primarily written for a somewhat technical audience that is only slightly aware of how different blockchains work. As such, it uses terminology and examples that are already common in Internet technologies and standard computer science. The explanation is intended to be neutral: neither promoting nor disparaging blockchain technologies.

The primary purpose of this document is to help the ICANN community understand current and proposed blockchain name systems, as described in Introduction to OCTO-039, "Blockchain Name System Technologies." It is organized as a verbose glossary that puts the various blockchain concepts into a technical and operational context.

Even though the current main use of blockchains is for finance, this document covers that use of blockchains only where necessary. A reader seeking information on the many ways blockchains are used can find it in various other publications about blockchains.

The terminology used in this document often does not match the terminology used by some in the blockchain ecosystem. In this document, the terminology is meant to be technology-neutral, whereas the terminology used in the blockchain ecosystems is often promotional, overly complex, or both. When this document uses terms different than are common in the blockchain ecosystems, it lists the similar terms from that ecosystem.

This is the first published version of this document. The author expects to publish a second version approximately six months after the first, to include corrections and additions that were received after the first version was published. This second edition will likely be longer, but will contain the same neutral tone. Please send all comments and questions about this document to octo@icann.org.

2 Basic Blockchain Components

There are thousands of different blockchains. This document offers a general overview of blockchain technology, but will sometimes reference specific aspects of Bitcoin or Ethereum. It gives an overview of the various parts of blockchain technologies, but is not meant to be comprehensive; a comprehensive document would be several times longer than this one, and would be out of date immediately on publication given blockchain technology's rapid pace of evolution.

The different components of blockchains are often interconnected. This interconnectedness makes writing a technical description of blockchains particularly difficult: it's unclear in which order to describe the components, and cross-linking the descriptions can lead to a tangled maze of references. The result is that the reader might not have a full picture of how the components interact until nearly the end of the document.

For example, when people talk about "the blockchain," they can mean a few different things (all described below):

- The list of transactions whose entries are cryptographically bound to each other

- The combination of that list with the rules that are needed to process the list
- The database that is the result of applying the list to the rules
- The combination of the transaction list, the rules, and the database

This document uses the term “blockchain” to be the latter: the whole system. Although the transaction list is the heart of the blockchain system, that list is useless without knowing the exact rules for processing the list. Further, the list component of the blockchain system is only useful for determining account balances and so on after it has been processed through the rules portion into the database.

2.1 Transaction List, Rules, and Database

The transaction list is an ordered list of transactions. A transaction is a direction to make some sort of change. A simple transaction common to nearly all blockchains is “transfer a coin from my blockchain account to this other account.” A more complex transaction might be “record some data in a database, and associate that data with my account.” An even more complex transaction that is available only in some blockchains is “run this program with these inputs” where the program can transfer coins, read data, and record data.

The rules are implicitly agreed to by everyone who participates in that blockchain’s particular ecosystem. This set of rules is sometimes called the *state machine* for the blockchain. In computer science, a state machine is a set of rules that is used to process a set of inputs into an unambiguous result called the *state*.

The result of running the list of transactions one-by-one through the rules is a database. Many people who care about a blockchain don’t care that much about the transactions listed in the blockchain itself, but instead about the contents of the database. That is, you might not care about all the transactions that led to a particular account having its current balance, but you care about the balance. However, you might also care about the historical aspect of the database, such as the list of transactions that affected a particular value in the database.

2.2 Coins

Talking about blockchains without any mention of finance would miss one of the foundational aspects of blockchains: coins. Many blockchains have a native coin associated with that particular blockchain. Blockchain coins create blockchain-based assets, often called “cryptocurrencies.” In the blockchain world, the terms “coin” and “token” are sometimes used interchangeably, but many people use “coin” to mean the specific currency of a blockchain and “token” to mean a currency not native to the blockchain. Many people avoid using the term “blockchain currency” because cryptocurrencies cannot be used in the same way as national or regional currencies because they have not gained significant acceptance as a form of payment for almost any items or services.

Cryptocurrencies may be similar to national currencies (such as the yen for Japan or the rupee for India) or regional currency systems (such as the Euro, the West African CFA franc, or the Eastern Caribbean dollar). A blockchain’s coin is used to pay for services, such as to pay for moving coins to other users of the blockchain, and for storing nonfinancial data in the blockchain’s database. Spending the blockchain’s coin is the primary way that many blockchains control who can add records to the blockchain.

The coins used in blockchains are also quite dissimilar to coins used by countries. Common blockchain coins (like Bitcoin for the Bitcoin blockchain and Ether for the Ethereum blockchain) are initially created by the creator of the blockchain itself and are not backed by the types of trust or commodities that back national or regional currencies. Some blockchains start with a fixed number of coins that cannot be increased. Other blockchains allow coins to be added to the total pool of those blockchains' coins based on some rules. Some less common blockchain coins are backed by traditional currencies or commodities.

Blockchain coins have names and abbreviations; some even have symbols. The names and abbreviations are not standardized, although they are informally agreed to by some financial organizations.

Coins and other aspects of blockchains and finance are covered near the end of this document.

2.3 Accounts

A blockchain allows multiple people or organizations to request transactions be added to the blockchain, such as payments with coins or to store data. Each entity that can add a transaction needs a separate identity so that someone reading the blockchain can find out who created each transaction. In this document, those identities are called *accounts*, but in different blockchain ecosystems they might be called "wallets" or "IDs" or other names.

Each account in a blockchain is based on a cryptographic key used for signatures. In specific, the account is based on the public key that is used to sign proposed transactions, and sometimes to sign other actions in the blockchain.

Blockchains often use short strings of text instead of the public key itself as identifiers. Such identifiers are usually human-readable representations derived from the public key, such as a truncated cryptographic hash of the public key that is rendered in text characters. In blockchains such as Bitcoin and Ethereum, these identifiers are called "wallet addresses", even though they are not addresses that point to a location (such as an IP address or a postal address).

The blockchain ecosystem uses the term "wallet" in two very different contexts. Sometimes it means the balance of coins that an account has just for the particular blockchain, but other times it means a collection of coins from different blockchains and other assets, such as those held at financial exchange services.

3 The Ledger at the Core of Blockchains

All blockchain ecosystems contain ledgers. Blockchain ecosystems use the term "ledger" for the ordered list of transactions in the blockchain. A common misconception among people with a technical background but little knowledge about blockchain technologies is that blockchains are just digital ledgers for coins. All common blockchains are more than that, although the value of the additional technologies used is not always clear.

A simple definition of a ledger is that it is an ordered list of items, where the entries in that list cannot be changed or removed after they have been added. (Technically, the ledger is a linked list, where an item's link is a cryptographic hash that points to the contents of the previous item.)

If you have a current copy of a ledger, you know that everyone else who has a current copy has an identical copy, with the same number of transactions and with every transaction being identical to the ones in your copy.

Ledgers can be added to, but only at the end of the ledger. Thus, you might have a copy of a ledger derived from fewer transactions than what makes up the current ledger, but even then, you can easily verify that the copy you have exactly matches the current ledger up to the point when your copy was made.

In a blockchain, the database that is created by running the items (transactions) from the ledger through the blockchain's rules is not a ledger, because it is constantly being modified as new transactions are being added.

In blockchains, each entry in the ledger is a *block* of transactions. The block is an ordered list of transactions, and because the ledger is an ordered list of blocks, that also means that the ledger is an ordered list of transactions, where the transactions happen to be batched into blocks.

3.1 Blockchain Ledger Example

Different blockchains have different types of ledgers, but many have common features. For example, the first block might be numbered 0 and the first transaction also numbered 0. The first three blocks might look like the following:

- The contents of the first block (called the *origin block*) might be special in that there are no transactions, but it still will have a digital signature.
- The next block, block 1, might have transactions 0 to 23, a cryptographic hash of the contents of block 0, and a digital signature from the account that put block 1 in the blockchain.
- The following block, block 2, might have transactions 24 to 34, a cryptographic hash of the contents of block 1, and a digital signature from the account that put block 2 in the blockchain.

3.2 Publishing the Ledger

Distributed ledger technology (often called *DLT*) is a method to widely distribute and authenticate the current contents of a ledger. Blockchains are probably the best-known example of DLTs, but there are many others. Blockchain ecosystems assume that anyone can get a copy of the blockchain's ledger (the list of transactions) and process them through the blockchain's rules on a local computer; this is one of the features that makes blockchains "distributed." The contents of DLTs can be distributed by web protocols (such as HTTP requests like "show me all the entries added after a specific time"); however, blockchains tend to use bespoke protocols that are optimized for distributing entries in rapidly expanding blockchains.

Many popular blockchains like Bitcoin and Ethereum don't have a universally agreed-upon ledger. Instead, at a particular time, everyone in the ecosystem agreed on the contents of the ledger up to a recent point, but did not yet agree on the last few entries in the ledger. The process of agreeing on what is in the up-to-date ledger is called *consensus*; the method for finding consensus is determined by each blockchain's ecosystem and is a defining feature of the blockchain's ecosystem (note that this use of the word "consensus" has a very different

meaning than in the community-based organizations such as ICANN or the IETF.) Other blockchain ledgers have predictable, formal rules for the full contents of the ledger because there is only one party that can write to the ledger. These concepts are covered in more detail later in the document.

The technology and cryptography for creating and publishing digital ledgers has been known for more than 30 years. Blockchains are a specialized form of DLT that has some common features related to publishing. These features, which may differ from earlier DLTs, include:

- Each entry in the blockchain is linked to the previous entry by including the cryptographic hash of the previous entry. This allows someone verifying the contents of the blockchain to unambiguously determine the order of the entries, to determine that no entries are missing, and to determine that no entries have been modified.
- Each entry can have a digital signature to allow someone verifying the contents of the ledger to know who added the entry to the blockchain. Blockchains require that the blocks are signed, and many require that the transactions are signed as well, depending on the trust model for the blockchain.
- The cryptography used in blockchain ledgers is almost always well-understood and follows long-established standards for long-term stability and resistance to attacks.

4 Getting from Transactions to Data

In order to determine something like “how many coins does this account on this blockchain have now” or “what are all the things that this account has written to the database,” you need to run the entire list of the blockchain’s transactions through the blockchain’s rules to create the database. This fact is fundamental to blockchain technologies, and also surprising to many people new to blockchains; only after you have created and filled the database by processing every block in the blockchain can you answer those queries.

4.1 Transactions

In many blockchains, most of the transactions are financial: “move this many coins from my account to this other account.” Another common simple transaction is writing to the database: “add this data to the database and associate it with my account,” “update this data associated with my account,” and so on. The variety and utility of transactions allowed by a particular blockchain are one of the most significant differentiators of different blockchains.

People or organizations with blockchain accounts create transactions that will be processed by those who put together the blockchain. When you desire a transaction to be part of a blockchain, you put together a transaction request in the format specified by the blockchain, and you apply your digital signature to the request. Requests must have a unique number in them (often an increasing integer) so that the same request cannot be added to the blockchain more than once.

Every blockchain needs well-defined methods to allow requested transactions to be added to the blockchain while avoiding the addition of unwanted transactions; often, there is a cost to even request a transaction be added to the blockchain (commonly known as a *gas fee*). Having too strict of a policy limits some potential uses of the blockchain, but makes the management of the request list easier. Having too liberal of a policy may allow requested transactions that are

spam (such as advertisements), malware (such as programs that drain accounts of their coins), and so on. Note, however, history has shown that transaction fees only reduce spam and malware, but don't eliminate it entirely. The choices an ecosystem needs to make for how to allow transactions into its blockchain are also covered later in the document.

4.2 Processing Rules and Programs

The rules for many databases are quite simple. Most blockchains have a rule that says something like "check the signature on this transaction; if it validates, move the specified number of coins from the account associated with the signature on the transaction to the account listed in the transaction".

Some blockchains (notably Ethereum and its derivatives) also support running computer programs. A program can be thought of as a set of blockchain rules rolled into one transaction. Thus, a transaction might be "run the program associated with a particular account, using these values as input to the program." These programs are often called "smart contracts" even though many of those programs are not at all contractual in nature.

Programs require a programming language, and the languages that can be run as a transaction vary between different blockchains. Typical blockchain languages are virtual machines controlled by opcodes, similar to opcode-based languages like Java or Python. Because these programs must run when the transaction containing the program is executed, running the same program with the same inputs at the same point in the list of transactions must always have the same result.

Blockchain languages must prohibit getting data from outside the database; examples of such prohibitions include not being allowed to get data from the Internet, not using random numbers, and not getting data from the physical computer on which the programs are running. However, many blockchain languages can read values from the database. For example, someone controlling a blockchain account might collect some relevant data from the Internet (such as the current exchange rates for various currencies) and use a transaction to write that data to the database. A transaction might run a program that reads that account's data from the database and use one of the exchange rates in the program. Programs that write data from the outside world into the database are sometimes called "oracles."

4.3 The Database Derived from the Ledger and Rules

The output of running the rules over the list of transactions is a database; that database might have multiple tables. Different people will want different kinds of reports from the database, such as "how many coins does this blockchain account have?", "which accounts have the most coins?", "what is the non-coin data associated with this account?", and so on. The format of the database is not specified, so two systems might have two very different databases, but the core information in the databases will be the same.

Current blockchain ecosystems tend to have the databases kept on the same hardware as the complete list of blockchain transactions. These systems are typically called *nodes* because not only do they collect copies of the blockchain and process it into the database. Nodes usually act

as a peer-to-peer network to distribute the ledger to new or existing nodes, but they also participate in creating new transactions and sometimes in adding blocks of transactions to the blockchain. For large blockchains like Bitcoin or Ethereum, a node requires a significant amount of computer hardware for storage and processing as well as a fast Internet connection to keep up with the stream of new transactions.

If you have your own copy of the database (such as if you are running your own node), you can trust your connection to that database. It is much more common, however, for you to not have a node but instead use the database from a service provider who runs a node and allows access to the database.

It is important to note that the structure of the blockchain itself disappears when creating the database. Fundamental features such as block numbers, signatures on blocks, and signatures on transactions do not appear in the database: only the results of the transactions do. Thus, the contents of the database are not secured by cryptography used in the blockchain; there is no standardized way to secure blockchain databases.

The size of the database depends heavily on the type of blockchain it is associated with. For a blockchain that only handles transactions about coins, the main table in the database might be quite small: the keys would be the account identifiers, and the values would be a single number. Other blockchains have databases that can be quite large, such as those that output the ledger of transactions, not just the summary of the accounts.

Database history tables will inherently be larger than the database itself, so it is up to the person running the blockchain rules how much history to keep. If too little history is kept, a query for a particular history might necessitate re-running all the transactions through the rules and adding a new table; if unneeded history is kept, the database will become large and possibly difficult to handle (such as making replication more costly). There are well-known methods for keeping database history in a relatively compact form, and this is used by some database maintainers who also do transaction research.

5 Putting Transactions into a Blockchain

There are two popular models for how requested transactions get added to the blockchain. In the *committer* model, there is just one party that adds to the blockchain. Everyone trusts that party to fulfill its role correctly and to only put transactions in the blockchain that will process successfully through the rules. In the *proposer* model (used by Bitcoin, Ethereum, and many others), there are many independent parties that suggest additions to the blockchain, and other parties (nodes) decide which of the blocks from those independent parties are actually part of the blockchain that creates the database. The two models are very different, and they originate from different models for trusting the ledger and database for a blockchain.

In both models, the parties adding to the blockchain have some tasks in common. They both must have a way of getting desired transactions from users. In the committer model, this might be a simple application programming interface (API). In the proposer model, users often send their desired transactions to a node that then transmits it to other nodes in an informal fashion. In the blockchain ecosystem, the committer model is sometimes called “centralized block producer(s)” while the proposer model is called a “decentralized block producing mechanism.”

When a committer is putting together a block of transactions, they need to check many things for each desired transaction. Depending on the blockchain, this might include considerations such as whether:

- The account proposing the desired transaction has enough coins to cover the cost of having the transaction published (such as having the transaction considered for publication, and the cost of running the transaction)
- The transaction request is well-formed
- The transaction has not already been published in an earlier block

5.1 Bad Blocks and Bad Transactions

A *bad block* is one that does not follow the rules for acceptable blocks; it can also be thought of as an invalid block. For example, if the signature on the block cannot be verified, that would be a bad block. A more likely example would be a block that contains bad transactions, such as those that would take an account's value negative, or that attempt to execute a program using the wrong types of input values, and so on.

The committer or proposer must create well-formed blocks with only well-formed and good transactions; otherwise, the transactions in that block will not pass all the rules. Bad blocks are normally rejected by the network of nodes before being committed to their database, but in cases where that doesn't happen, a node needs to back out transactions not only from the bad block but from all blocks accepted after that block. The possibility of bad blocks leads to the fact that all ledgers are not actually immutable: a ledger is only immutable up to the block before the first bad block. One possible rule for processing a bad block is "never trust the blockchain again after this point," but a much more common processing rule is "ignore the bad block's existence and pretend that it never appeared." Different blockchains have different rules for when the data in the database can be considered immutable; for example, Bitcoin generally assumes a block is valid after six blocks have been accepted after it.

5.2 The Committer Model

In the committer model, all rules for what goes into the blockchain are made by a single committer. The committer gets to decide who is or is not allowed to submit requests, whether there is a limit on the number of requests, how and how much the party that is requesting their transaction to be added to the blockchain needs to pay, and so on. They also get to decide the timing of addition of blocks to the blockchain, such as whether they are added at regular increments or when a certain number of transactions are ready. Each block added to the blockchain is signed by the committer using a public key that is somehow trusted by all that blockchain's users.

In the committer model, no one needs to be paid to add blocks: the one committer does this based on whatever their business model might be. Many blockchains that use this model have a simple pay-to-commit scheme: you pay (either blockchain coins or traditional money) to get your transactions accepted in the blockchain.

Typically, users with desired transactions send their requests to the committer. The committer can look through the list of desired transactions, putting them into a block in an order defined by

the committer or some rule defined by the ecosystem, and possibly communicate with users when their blocks are accepted or rejected.

This model, while simpler than the proposer model, has some fragility based on the single point of failure. There may be legal, technical, or emergency reasons why the committer cannot continue to add blocks in a timely fashion. If there is no replacement for the committer, the blockchain may become unusable or untrusted. Blockchains using the committer model can have rules that allow transition of the committer role to other parties, such as transactions that would specify the public signing key of the new committer.

5.3 The Proposer Model

In the proposer model, many entities can attempt to add blocks to the blockchain. These entities are often called “miners” and “validators” in blockchain ecosystems. Proposers are generally not coordinated with each other, and thus two or more proposers might create a new block for the blockchain to follow the same existing block. (Some blockchains have organizations that run many parallel proposers at the same time in order to make more money by having more blocks accepted by the blockchain.) It is up to the nodes to decide which competing block is part of the long-term blockchain.

The proposer model leads to an inherent problem for users of the blockchain: when can they trust a recently added block is valid according to the blockchain’s rules? It is easy to imagine that two proposers could propose new blocks, and nodes following those proposers can get even more blocks following the two different paths. Before any nodes decide to update their database, they have to believe that they are using blocks that others in the ecosystem will use. This is the heart of the issue of consensus described earlier: there is no firm rule for when a node will trust a new block, even if there are other blocks added after that block. Different blockchain ecosystems use different guidelines for making that decision. For example, a node in Ethereum might wait until two or more additional blocks appear after a new block before that node “finalizes” the block by processing it through the rules into the node’s database. In Bitcoin, nodes often wait for six or more subsequent blocks.

The proposers can collect desired transactions from a single pool shared by all proposers or, more typically, each proposer has their own pool that they receive from accounts with desired transactions. The proposers might share their pools, depending on the rules established by the blockchain’s ecosystem. When a node sees that a desired transaction has been added to a block in the blockchain, it has to remove that desired transaction from its pool.

In the proposer model, proposers who successfully add a block to the blockchain are rewarded, typically with the coins of that blockchain. A proposer looking though their pool might choose particular transactions for an upcoming block if including those transactions might benefit the proposer more than others (such as due to higher fees offered to include the transaction), or appear to be helpful to the overall usefulness of the blockchain. Some proposed transactions might never be added to the blockchain if no proposer wants to add it when they are creating blocks.

Blockchains that have multiple proposers have rules for deciding which proposer gets to add the next block. Without such rules, the ecosystem would be inundated with proposed blocks, which would make coming to consensus significantly more difficult. Two of the best-known sets of

rules for choosing who among a set of proposers can post the next block are based around the ideas of *stake* and *waste*.

Stake: Holding a substantial amount of a blockchain's coins can be an indicator of your commitment to the blockchain and an incentive to be a good actor in the blockchain. This can in turn be considered an indicator of your trustworthiness to add blocks that will be accepted into the blockchain.

Waste: The ability to waste resources, particularly electricity and computing power, has been a popular mechanism for allowing someone to be a proposer since Bitcoin originated the idea. This method is commonly called "proof of work" even though there is no useful work being done. That is, the resources are being expended without having a direct effect on the blockchain; they are only used to signify the desire to be able to publish blocks and be rewarded for such publication by receiving coins. Potential proposers compete to show who can waste the most the fastest in order to win the competition.

6 Trust Models for Blockchains

When you rely on any source of data such as a database based on a blockchain, you inherently must have some level of trust in the authenticity of those who have entered the data or in the nodes validating the data. You also need to have some level of trust in the integrity of the data (that is, that the data appears the same as those who entered it intended it to be). Those who provide the data have established practices that give you more or less trust in the authenticity and integrity of the data, and you choose how much to believe in those practices. This is called the *trust model* for the data.

Different DLTs have a wide variety of trust models, and it is often the trust model that causes some blockchains to attract different sets of users. Nearly every blockchain has the same trust model for data integrity: ordered blocks that have the cryptographic hash of the previous block embedded so that the integrity of the whole chain is easy to verify.

However, the question of how to trust the authenticity of the transactions in each blockchain is a central concern for that blockchain ecosystem. For blockchains, a significant part of the trust model is that you can fully trust that the transactions in the blockchain, and that all of the following are true:

- Every transaction was requested by purported account holder
- The blockchain contains every intended transaction
- Every transaction, when run through the blockchain's rules, will be a valid change to the database

In the committer model, if the entity that writes the blockchain makes a mistake, such as adding a transaction that cannot be processed due to insufficient funds in an account, you might lose trust in the entity and the blockchain becomes unusable to you. Similarly, in the proposer model, if a set of nodes show a recent change to an account to a new amount, and then revert that change due to a changed consensus, you might no longer trust those nodes. Thus, the trust model is not just about "is the data valid" but also "what is the chance that the data is authentic but still incorrect".

This section gives just an overview of the complex decisions that go into creating a sensible trust model that can be widely believed.

6.1 Trusting the Origin Block

Every blockchain has to start somewhere. The first block of a blockchain has no previous block to reference by including its cryptographic hash. Thus, users of a blockchain must somehow trust that the first block is the authentic start for the blockchain. The technical term for this type of trust is a “leap of faith.”

A trust model for new blockchains that are not yet well established needs to include how to trust the beginning of the blockchain. There are two main ways to make this leap of faith. If there is a known entity that takes credit for creating the first block, and that entity has a well-known public signing key, they sign the first block. If there is no such entity, everyone trusting the contents of the blockchain must simply trust that the first block was created by someone they would have wanted to start the blockchain.

6.2 Trust Model for the Committer Model

A single source of data is the simplest to implement, the simplest to understand, and has the simplest trust model. The trust model is all-or-nothing with a single point of success or failure. You can trust accessing the database if it is provided by the same entity that created the blockchain. If you want to trust someone else to create and run the database for the blockchain, you have a high assurance that their database would match that of anyone else because they are all getting the definitive copy of the blockchain.

A wrinkle in this trust model is that the more complex the process rules for the blockchain are, the more likely it is that different software processing the blockchain might end up with different records in the database. To avoid such divergence, all entities that process the blockchain might use the same software, but this introduces even more fragility. Alternatively, the blockchain could have a test suite for database creation.

6.3 Trust Model for the Proposer Model

Trusting two or more proposers who each have equal privileges is significantly harder than trusting just one committer. The trust model shifts from you trusting the entity that is creating blocks to you trusting a particular node or set of nodes that write the blocks into the database. Without a useful rating system for trustability of nodes, you are left with guessing which node or nodes would be best for your queries, or running your own node and trusting that you can do so well.

A second complexity in the proposer model that is not in the committer model is if you want to submit transactions. In the proposer model, you need to choose one or more nodes to submit to; without a useful rating system, you may end up submitting to a node that ignores you or puts your transactions at a disadvantage to others.

An oft-stated reason that many blockchains have multiple proposers is to avoid the likelihood that some proposers go rogue and start committing transactions that help them financially. A

similar reason is that a single proposer might prevent a valid transaction from getting added to the blockchain for political or financial reasons, but with more proposers, such transactions are less likely to be blocked. This is often part of the discussion of “decentralization,” but having many proposers doesn’t change the ability for you to trust the data in a blockchain: unless you are running your own node, you must still trust the honesty of the node you are sending queries to.

The discussion so far is about the trust models for people using a blockchain’s database. There are also trust models for other parts of the proposer model, such as how the nodes would trust each other to know when there is consensus that a particular block is to be processed into the database. There are many ideas in the blockchain ecosystem for different consensus mechanisms where some of the variables include speed to agreement, cost of communications, efficiency of backing out of wrong guesses, and so on.

7 Cryptography Used in Blockchains

Nearly all blockchains use cryptography in similar ways; this is fortunate for those trying to understand blockchains because it reduces the amount that needs to be learned when comparing different blockchains. Two cryptographic primitives, hash algorithms and digital signatures, are used in many places in blockchain technologies. This document does not explain the cryptography itself, just how it is used to make blockchains secure against attacks. (There are many books that explain modern cryptography in depth; a good place to start is *Serious Cryptography*, 2nd Edition, by Jean-Philippe Aumasson, ISBN 9781718503847.)

Note that the cryptography described here is for the blockchain’s transaction list, but not for the database that is the result of processing the transaction. The blockchain ecosystem has no standard for securing blockchain databases.

7.1 Hash Algorithms

Cryptographic hash algorithms are what proves that the ledger is complete and has not changed over time. As described earlier, each block in a blockchain contains the hash value of the previous block and a set of transactions.

You can verify that a ledger has not been modified by checking that the hash value listed in each block is the hash of the block just before that one. Starting from the origin block, at any particular block, if the hash of the previous block verifies, you can be sure that all the previous blocks were unmodified.

7.2 Digital Signatures

Digital signatures are used to authenticate transactions in a blockchain. Blockchains typically use elliptic curve cryptography for their digital signatures.

When a transaction is proposed to a blockchain, it needs to be signed by the key associated with an account in the blockchain. This transaction signature verifies the authenticity of the proposed transaction. The public key used in the signature is used to determine the account responsible for the components of the transaction, such as where coins should be transferred

from. For example, a transaction that says “give five coins to that account” would cause those coins to be removed from the account associated with the public key signing the transaction.

Digital signatures are also used to limit who can add blocks to a blockchain in the committer model.

8 Associations Between Blockchains

The two best known blockchains, Bitcoin and Ethereum, have limits on how fast blocks can be added largely due to their popularity and competition to be proposers; in fact, for Bitcoin, it is part of the technical design. These blockchains have fluctuating costs for adding transactions, and varying speeds at which there can be agreement across databases, which also limits their utility in many scenarios.

Because of this, the blockchain ecosystem has created a large number of additional blockchains built on top of Bitcoin and Ethereum for scenarios that cannot be fulfilled by Bitcoin and Ethereum. These blockchains are called *L2 chains* (“L2” is the common shorthand for “layer 2”). L2 chains are those that run independently of higher-level *L1 chains* (“layer 1” chains) but interact with the L1 chains in some way. The best-known L1 chains are Bitcoin and Ethereum, but there are other L1 chains that are gaining in recognition and popularity.

Some blockchains are independent of other blockchains; these are called *sidechains*. Sidechains might have many of the technical and trust properties of particular L1 or L2 chains (but with different origin blocks), or some have very different properties that are meant to have different use and trust profiles from existing blockchains. If a sidechain becomes important enough, and if it has scaling or convergence problems, the blockchain ecosystem might consider it an L1 chain.

A common use for L2 chains is faster and less expensive coin transactions of Bitcoin or Ethereum. An account on an L1 chain will transfer some coins to an account on an L2 chain, and those coins are then part of (faster and less expensive) transactions on the L2 chain. At some point, the coins in the account on the L2 chain are moved back to the L1 chain. L2 chains tend to be highly centralized. L2 chains are generally solving the problem of scaling in an L1 chain by adding trust calculations and higher risk in exchange for better overall throughput of transactions.

Another common use case for L2 chains is to support types of transactions not supported on the L1 chain, and to periodically record a cryptographic hash of the L2 blockchain to an L1 chain. Doing this type of recording is meant to increase the trust in the L2 chain for people who trust the L1 chain’s integrity more than that of the L2, although it might be difficult to quantify that additional trust.

There are many other proposed uses for L2 chains, but the technology and trust model for them have not been clearly documented. There is little agreement in the blockchain ecosystem about when a blockchain is really an L2 chain, despite the high amount of interest in L2 chains. This is due in part to the idea that, when an asset or block of data moves off of the L1 chain on to an L2 chain, the view of the database for the L1 chain is that the asset has disappeared or is frozen for reasons not clear to the database. In the future, the linkages between L1 and L2 chains will possibly become more formal, and thus the associations will be stronger.

L2 chains are always aware of the L1 chains they are associated with. There are newer L1 chains that are aware of associated L2 chains. In this model, the L2 chain can act as a faster transaction cache that the L1 chain can pull in automatically.

9 Blockchains and Finance

The purpose of this document is to help readers interested in blockchain technology, particularly when used for blockchain name systems. Even though finance is what interests today's users of blockchains more than anything else, the use of blockchains for finance is not central to the technologies used in blockchains; instead, finance mostly affects the trust models described earlier in this document.

9.1 Coins

Many blockchain coins have named fractions, similar to the way that the United States currency is discussed both as "cents" and "dollars." For example, the Ethereum coin is called *Ether* (informal symbol: *ETH*), but it is common to see transactions of Ethers in units of *wei* and *gwei*, where there are 10^18 wei in an Ether and 10^9 gwei in an Ether. In the Bitcoin ecosystem, it is common to see transactions in units of *satoshis*, where there are 10^8 satoshis in a Bitcoin.

9.2 Additional Assets

The blockchain ecosystem has spurred the creation of additional types of assets associated with blockchains. Probably the best known such asset is the *non-fungible token*, better known as the *NFT*. An NFT is simply a record in a database that one account claims that another account owns as a particular asset (the token). The difference between an NFT and a blockchain coin is that each NFT is unique and that it cannot be subdivided.

Typically, an NFT is a digital file (an artwork, a piece of music, some writing, and so on), a URL that resolves to that file, and an entry in a blockchain's database to that URL. Some in the blockchain ecosystem have a narrower definition of an NFT, namely that only the transaction in the blockchain is the NFT. Sometimes, the NFT comes with a transfer of other rights (such as copyright or republication rights), but such transfers are created off the blockchain due to limits of what can be expressed on blockchains and agreement on the legal implications of such expressions. Selling the NFT entails transferring the blockchain entry from one account to another, often in exchange for cryptocurrencies or other NFTs.

OCTO-039, "Blockchain Name System Technologies" talks about blockchain name systems. A hierarchical name that is part of a blockchain database is a kind of NFT: each name is unique within its hierarchy, and a single name cannot be subdivided. Names in the global DNS are not currently controlled in blockchains, but there is some interest in parts of the ICANN community in doing so in the future for some names.

10 Conclusion

Blockchains are based on a wide range of technical choices for their fundamental technologies. The design of an individual blockchain includes a mix-and-match selection of trust models,

ledger mechanics, relationship to other blockchains, and financial objectives; of course, there are plenty of non-technical design choices as well. Understanding the technology that underpins a blockchain design that is being used for a particular purpose (such as a blockchain name system) will help you evaluate the fitness of the design and to compare it to other designs of other systems that might have similar purposes.