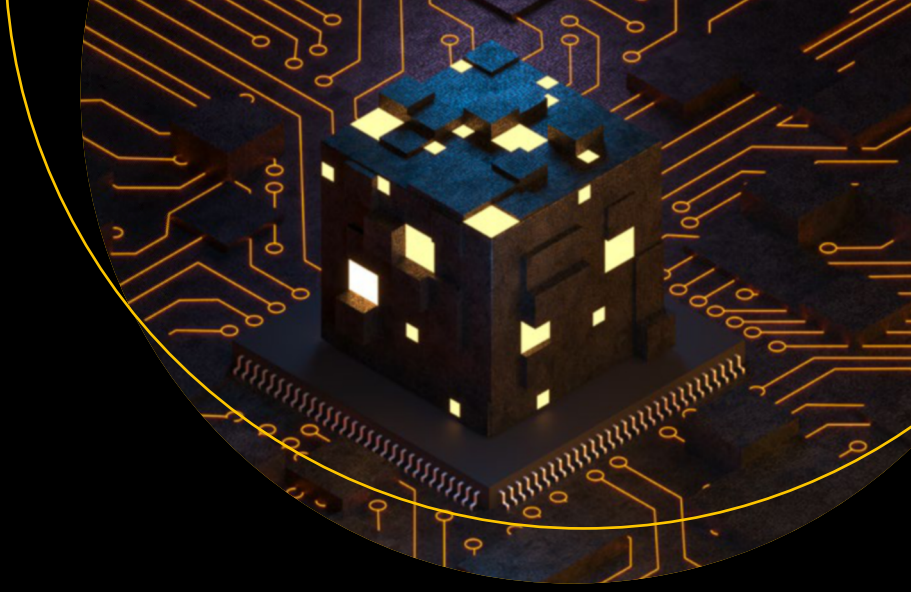

Testing the Creditcoin Blockchain

A Daily Account from a Test Engineer's Perspective

Alexander Todorov

Apress®

Testing the Creditcoin Blockchain

A Daily Account from a Test Engineer's Perspective

Alexander Todorov

Apress®

Testing the Creditcoin Blockchain: A Daily Account from a Test Engineer's Perspective

Alexander Todorov
Sofia, Sofiya, Bulgaria

ISBN-13 (pbk): 979-8-8688-0872-2 ISBN-13 (electronic): 979-8-8688-0873-9
https://doi.org/10.1007/979-8-8688-0873-9

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Malini Rajendran
Desk Editor: James Markham
Editorial Project Manager: Gryffin Winkler

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, USA. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub. For more detailed information, please visit https://www.apress.com/gp/services/source-code.

If disposing of this product, please recycle the paper

To the tester who gave me everything.

You showed me what true, unconditional love and sacrifice is. I will always cherish the time that you allowed me to be together with you!

I am deeply sorry that I was too selfish to not realize it in time. I apologize from the bottom of my heart that I hurt and betrayed you by taking away your most sacred dream.

Please forgive me!

Table of Contents

About the Author xi

About the Technical Reviewer xiii

Acknowledgments xv

Preface xvii

Part I: Introduction to Blockchain 1

Chapter 1: Introduction 3

Introduction to Blockchain 4

Blockchain Attributes 6

Blockchain Glossary 8

Blockchain Components 17

Peer-to-Peer Networking 19

Consensus Algorithm 19

Transactions 20

Transaction Fees 21

Block Production 21

Block Rewards 22

Remote Procedure Call (RPC) API 22

Storage 23

Migrations 24

Runtime Upgrades 24

Metrics and Telemetry 25

Off-Chain Worker or Oracle....25
Smart Contracts....25
Consensus Mechanisms....26
Proof of Work (PoW)....27
Proof of Stake (PoS)....27
Proof of Authority (PoA)....28
Summary....28

Chapter 2: Blockchain Development Frameworks....29

Hyperledger Sawtooth....29
Substrate....30
Ethereum....31
Programming Languages....31
How Do You Build a Blockchain Implementation....35
Thoughts on Development and Testing....38
Summary....40

Part II: The Creditcoin Blockchain: Aims and Objectives....41

Chapter 3: Introducing the Creditcoin Blockchain....43

What Is Creditcoin....43
Impact on Testing Activities....47
Summary....50

Part III: Testing the Creditcoin Blockchain....51

Chapter 4: Creditcoin 1.x....53

Components of Creditcoin 1.x....55
Client....55
Validator....56
Consensus....57

Transaction Processor(s) 58
REST API 59
Gateway 60
SDKs 61
Creditcoin-Legacy-Tests 61
Creditcoin-Legacy-Docker-Compose 62
Creditcoin-Legacy-Docker-Compose-Testnet 62
Timeline of Creditcoin 1.x 62
Testing Creditcoin 1.x 63
State of Testing Before I Joined 66
Testing Overview 66
Improvement on Tests 69
Testing v1.7 ➤ v1.8 Switchover and Suspected Networking Issues 74
The End 75
Summary 76

Chapter 5: Creditcoin 2.0 79

Components of Creditcoin 2.0 81
Extrinsics Pallets 83
Transaction Fees 85
Weights and Benchmarks 86
Runtime 86
Storage Migrations 87
Custom RPCs 88
Telemetry and Custom Metrics 88
Creditcoin-js 89
Creditcoin-squid 89

Timeline of Creditcoin 2.0 90
Testing of Creditcoin 2.0 91
Unit Testing 93
Integration Testing 98
Sanity Testing and Static Analysis 102
Testing with Bots 104
Migrations and Upgrade Testing 106
Continuous Testing on Pull Requests, Devnet, Testnet, and Mainnet 112
Security-Related Testing 118
Testing Creditcoin-squid 120
How We Found a Bug at Block 1 Million 120
Other Interesting Facts 121
Summary 124

Chapter 6: Creditcoin 2.3 125

Components of Creditcoin 2.3 126
Staking-Related Pallets 126
Creditcoin-cli 126
Switch_to_pos() 127
Block History 129
Creditcoin Staking Dashboard 129
Subscan Essentials 130
Creditcoin-squid 130
Timeline of Creditcoin 2.3 130
Testing of Creditcoin 2.3 131
Unit Tests 131
Integration Tests 132
Testing Subscan Essentials 134
Testing Creditcoin Staking Dashboard 135

Testing Creditcoin-cli 136
Testing Gluwa's Substrate Fork 138
Community Testing on PoS Testnet 140
Documentation Review 144
Load and Performance Testing 145
Security Bounty Program 147
Other Interesting Testing and Some Bugs 149
Testing Challenges During v2.3 154
Migration from PoW to PoS 157
Summary 157

Chapter 7: Creditcoin 3.0 159

Components of Creditcoin 3.0 160
Polkadot-sdk 160
Frontier 161
Creditcoin3 162
EVM Tracing RPC 162
Precompile(s) 163
Creditcoin 3 CLI 164
Proxy Functionality 164
Staking Dashboard 166
Subscan API 167
Blockscout 167
Crunch 168
Timeline of Creditcoin 3.0 168
Testing of Creditcoin 3.0 168
Unit Tests 169
EVM and Smart-Contract Testing 170
Testing EVM Tracing Functionality 171

Integration Tests for the Blockchain 172
Precompile Testing 173
Testing Creditcoin 3 CLI and Proxy Functionality 174
Runtime Upgrade Testing 185
Testing Creditcoin Staking Dashboard 185
Testing Gluwa's Polkadot-sdk Fork 189
Testing Gluwa's Frontier Fork 189
Testing Gluwa's Crunch Fork 190
Documentation Review and Third-Party Tools Testing 190
Other Interesting Testing 192
Summary 206

Part IV: Blockchain Testing Approaches 207

Chapter 8: How Others Test Blockchain 209

My Blockchain Testing Approach 209
Testing Across the Blockchain Stack with Andrew Snaith 211
Products Under Test 212
Testing Strategy 214
Testing Smart Contracts with Sebastian Viquez 222
Product Under Test 223
Testing 226
Summary 241

Index 243

About the Author

Alexander Todorov is a quality assurance engineer and an ISTQB and Red Hat certified professional with two decades of industry experience, an open source hacker with thousands of contributions, a Python instructor, and co-organizer of FOSDEM's Testing and Continuous Delivery devroom. He's also tested multiple components of Red Hat Enterprise Linux, was a test lead for the entire lifecycle of its version 5 family, and later tested various components of the Creditcoin blockchain.

Alex holds a master's degree in Computer Engineering from the Technical University of Sofia, Bulgaria, and loves public speaking and riding fast motorcycles. In his spare time, he's the lead engineer behind Kiwi TCMS, a popular open source test management system.

You can find Alex on LinkedIn, https://www.linkedin.com/in/alextodorov/, and follow his work directly on GitHub via https://github.com/atodorov.

About the Technical Reviewer

Anandaganesh Balakrishnan is a data engineering and data analytics leader who has held senior leadership roles across the fintech, biotech, and utility domains. His expertise is architecting scalable, reliable, and performant data platforms for advanced data analytics, quantitative research, and machine learning. His current research is AI on unstructured data, large language models (LLMs), generative AI (Gen AI), self-service data analytics, and data catalogs.

Acknowledgments

I want to thank Jeroen Rosink for being the first one who gave me feedback about the contents of this book. You've given me valuable feedback several times till now. I may have misunderstood some of it and definitely not incorporated everything, but I have no doubt that your feedback has made a positive difference on my end.

Next round of thanks goes to Sebastian Malyska who helped me get in touch with other blockchain testers. Seba, I hope we meet soon at another testing conference!

Thank you to my fellow tester and neighbor Liviu Damian for putting me in touch with Andrew Snaith.

Thank you Andrew Snaith for agreeing to share your testing experience for this book which added many other dimensions to my personal stories.

Last but not least, thank you Sebastian Viquez for agreeing to share your testing experience around smart contracts which completes this book.

Preface

This book follows the quality engineering journey of the Creditcoin blockchain across four distinct implementation versions and a myriad of technologies from a first-person point of view. It discusses testing implementations with the Hyperledger Sawtooth and Substrate blockchain frameworks, testing switch from proof-of-work to proof-of-stake consensus algorithm and testing an Ethereum Virtual Machine compatibility layer.

Readers will traverse several years of fast-paced blockchain implementations and technological changes including explanation of all major components under test, the approach taken, including examples of test automation and tools, interesting bugs, and testing challenges. Chapters follow each major implementation of the Creditcoin blockchain and conclude with a couple interviews with other testers working on different blockchain-related products.

Almost 99% of everything discussed in this book is open source, and multiple references to source code and GitHub are included throughout.

This book is for the software tester and quality engineering folks, who may soon be working on a blockchain implementation without having the necessary understanding of what it is, how it works, and what is important in terms of "assuring" quality.

Target audience:

- Software tester
- Software developer in test automation
- Quality assurance engineer
- Quality assurance manager

What you will learn:

1. Key components of a blockchain with some diagrams
2. Glossary of popular blockchain and crypto terms
3. Overview of blockchain implementation with the Hyperledger Sawtooth framework
4. Overview of blockchain implementation with the Substrate framework
5. Practical topics related to testing proof-of-work, proof-of-stake, and EVM-based blockchains

PART I

Introduction to Blockchain

CHAPTER 1

Introduction

Cryptographer David Chaum first proposed a blockchain-like protocol in his 1982 dissertation "Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups" with further improvements on this idea in the next 15–20 years.

The first decentralized blockchain was conceptualized by a person (or group of people) known as Satoshi Nakamoto in 2008 giving rise to Bitcoin. 2013 was the start of Ethereum which introduced smart contract functionality. Due to concerns over the high energy consumption required for cryptographic computations, later chains such as Cardano (2017), Solana (2020), and Polkadot (2020) started adopting the less energy-intensive proof-of-stake model.

At present, we are witnessing continuously increasing interest and investment into blockchain technology[1] with a multitude of companies searching for practical applications beyond cryptocurrencies. It is the time when blockchain technology is entering the world of the more common software engineering folks. It is also the time when blockchain technology is being productized, which leads to the necessity of more testing for these products.

As an experienced quality assurance engineer with nearly two decades of experience, I felt very intrigued about working in the blockchain testing space, yet at the same time, I felt extremely uncomfortable for a very long time. I've had prior experience with testing low-level infrastructure type

[1] https://www.statista.com/topics/5122/blockchain/#statisticChapter

A. Todorov, *Testing the Creditcoin Blockchain*, https://doi.org/10.1007/979-8-8688-0873-9_1

of software, like various parts of the Linux operating system, but didn't have any experience with cryptocurrency or blockchain before. Because I was working on a product that was already in production, I needed to learn a lot of things very quickly in order to be a productive member of the team. It was like drinking from a firehose, and to this day it feels like that sometimes.

This book is for the common tester and quality engineering folks, who may very soon be required to work on a blockchain software product without having the necessary understanding of what it is, how it actually works, and what is important in terms of "assuring" quality.

Blockchain networks are distributed systems which are usually more complex compared to regular software products, and there are a myriad of items to be tested which requires a vast skill set. This book is not meant to be an exhaustive resource for every possible scenario; that is impossible. Instead, it is meant to tell my story as a blockchain test engineer and hopefully be your starting point in this domain.

Happy testing!

Introduction to Blockchain

There are multiple components in distributed systems like blockchain, and some of them are not necessarily common knowledge unless you have worked in this specific technical field before. This chapter explains what a blockchain is and establishes some of the common terms used in this domain and gives you some examples.

A blockchain is a growing list of records, called blocks, which are securely linked together by cryptographic functions which guarantee that data is tamper proof. In the context of Figure 1-1, we can think about each block as a snapshot of the current state at a specific period of time. If someone tries to modify this state at a later date, that would cause all of the blocks afterward to be invalid with respect to the modified block,

thus revealing data tampering. It is also referred to as a distributed digital ledger. Such architecture allows for participants who don't trust each other to still have trust in the data recorded onto the blockchain.

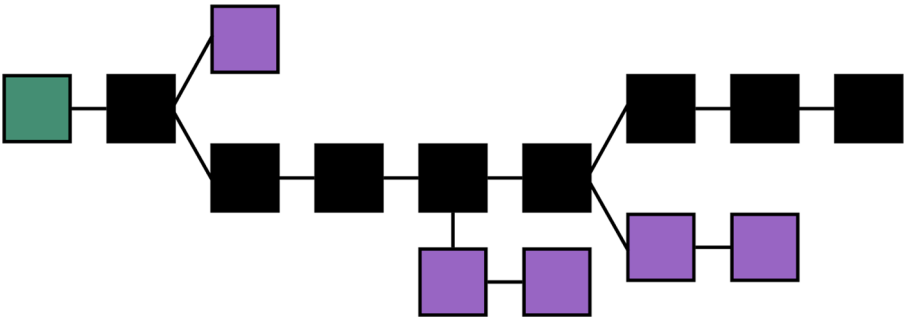

Figure 1-1. *Example of linked data blocks forming a blockchain License: CC BY 3.0, source* https://commons.wikimedia.org/wiki/File:Blockchain_landscape.svg

Blockchain participants are individual operators who execute the blockchain client software on their computer systems. Their motivation is usually financial – to gain rewards for providing distributed computing resources to the blockchain.

In order to record new information onto the blockchain, participants must be in agreement – for example, what is the answer to a difficult mathematical problem. This is known as consensus. By design, most blockchain systems are open for participation to anyone, and in many cases participants are anonymous, could be faulty (due to software or hardware bugs) or misbehaving (due to possible external incentive to falsify records in the blockchain), and generally cannot be trusted. It is the job of a consensus algorithm to ensure that participants in such a system can reach an agreement and to detect and exclude participants who are considered to be at fault. Otherwise, the blockchain software cannot proceed to the next step, which is recording actual data into storage.

Once new transactions are combined into a block and this block is distributed across multiple nodes in the blockchain, it cannot be erased or modified. The blockchain ensures full transaction history, and the use of cryptographic functions ensures that information recorded on the chain is secure by design. These records are essentially read-only. Algorithms and software architecture are chosen so that they guarantee these principles. The actual cryptographic, consensus, networking, and other algorithms and implementation architectures vary, and there is a lot of ongoing research in this space, but the underlying principle is ensuring data integrity as shown in Figure 1-1.

Although many of today's blockchain platforms are capable of supporting smart contracts and cryptocurrencies, these are not strictly required for the core operation of a blockchain as a secure distributed ledger. It is the opposite, being a secure distributed ledger, that makes the existence of smart contracts and cryptocurrencies possible.

A blockchain implementation is an infrastructure type of software which on a very high abstraction level resembles a database – there is some data recorded in storage and the outside world, that is, other applications can read and write to the blockchain.

Blockchain applications are usually more complex than other pieces of software that testers may be familiar with, with multiple components working in parallel and multiple networking channels and protocols being active at the same time.

Blockchain Attributes

This section is included only for your reference. In a practical implementation, most of these attributes are available to you out of the box, courtesy of the chosen blockchain framework.

The testing requirements around these particular attributes depend on your operational and business requirements and in practical implementations, for example, where you are not developing the underlying algorithms themselves, are most likely going to be resolved by having heavy monitoring and alerts rather than a functional test suite.

- **Security**: Information on a blockchain is generally considered to be secure against modification so that historical records can be trusted. The blockchain is also considered generally secure against malicious actors because of its distributed nature
- **Fault tolerance**: In a blockchain network, there may be nodes which exhibit faulty behavior or are downright malicious. Blockchain implementations have built-in tolerance against this.
- **Reliability**: The probability that the blockchain will work properly in a specified environment and for a given amount of time. For nominated proof-of-stake consensus, this translates to having at least two-thirds of all elected validators online and functioning properly during the current era.
- **Reproducibility**: Transactions on the blockchain can be replayed by another node which should always result in the same state, for example, same information stored locally on disk for the node replaying the transactions.
- **Transparency**: Records on a public blockchain are accessible to everyone, and all state transitions, from genesis to the current block, can be replayed and verified individually. That is, even though participants are generally anonymous, all of their actions are fully transparent and can be retraced by anyone.

- **Finality** is the level of confidence that a well-formed block recently appended to the blockchain will not be revoked in the future and thus can be trusted. Most distributed blockchain protocols cannot guarantee the finality of a freshly committed block, and instead rely on "probabilistic finality": as the block goes deeper into a blockchain, it is less likely to be altered or reverted by a newly found consensus.
- **Public vs. private**: Whether the blockchain is fully open to the public, for example, anyone can participate or not. Blockchains that you hear about in the news are public, including Creditcoin.
- **Permissionless vs. permissioned**: Whether anyone can join an existing blockchain network, operate a node, and participate in consensus with/without first obtaining some sort of permission, approval, etc. Popular blockchains like Bitcoin, Ethereum, and, of course, Creditcoin are permissionless.

Blockchain Glossary

Transaction much like in a regular SQL database is an operation that seeks to append new information onto the blockchain. It is usually exposed as an API function so that users may interact with the blockchain. Blockchain transactions are usually a singular entity analogous to a single SQL statement executed against a traditional database. Transactions represent the blockchain's business domain, for example, `AddBidOrder` for Creditcoin or transfer of Ether from one party to another in the case of Ethereum. There can be multiple transaction types supported by a blockchain network.

A **hash** is the result of a one-way cryptographic function usually receiving binary data as input and returning a hexadecimal string! Each data has a unique hash representation, and you cannot reverse the hash value back to the original input data. Even a one-bit modification of the input data will result in vastly different hash value. This makes it safe to publicly share the hash value in lieu of the data itself and to use the same hash value to verify that two pieces of data are exactly the same thing.

Transaction hash is a hexadecimal string representing the cryptographic hash of each transaction. Very often these strings are used in lieu of identifiers. For example, `https://etherscan.io/tx/0x8d116ea41628ac89745c95147c0ea394809074607395b0d792d1bea9136280b1` represents a transaction where I have exchanged one crypto token (G-CRE) for another (ETH).

Gas or **transaction fee** is the representation of how much it costs to execute a particular transaction on the blockchain. It can vary over time and between transaction types and is meant to represent the computational expense for the transaction. This value is usually represented by the blockchain's native crypto token. The purpose of gas is related to token economics and also provides a security feature making malicious behavior economically not viable.

A **block explorer** is typically a web application which allows the user to search and inspect blocks, transactions, and addresses on a blockchain. A popular one is Etherscan linked in the previous paragraph. For Creditcoin, a similar function is performed by Subscan (`https://creditcoin3-testnet.subscan.io/`) and Blockscout (`https://creditcoin-testnet.blockscout.com`).

Block is a collection of multiple transactions that are recorded atomically onto the blockchain. Either all transactions within the block are recorded together or none of them are. If we make an analogy with the git version control system, a single chunk of modified source code will be a "*transaction,*" while a git commit of multiple modifications will be "the *block.*"

While we may commonly refer to blocks as records, the meaning is different compared to a database record, which represents a singular collection of discrete values, in other words a row of column values. In the blockchain context, the term record is closer in meaning to the vinyl disk record which holds multiple songs.

Blocks contain a list of transactions, references to the previous and next blocks, as well as additional metadata such as time stamps, cryptographic hashes, and signatures. There is a maximum limit of how many transactions can be stored in a single block, depending on implementation.

Block fullness or **block saturation** level represents the percentage of how much of the total available space within a block is actually used. For example, if the blockchain is not in use, each block may hold a single heartbeat/timestamp transaction. Or it may hold hundreds of transactions in a heavily used blockchain.

Block hash is the hexadecimal string representing the cryptographic hash of each block. Very often these strings are used in lieu of addresses or identifiers. For example, `0x34e89f72d1bc09af19759a94c347d769a94bb0ca9d257b2efe8ad5ac0502cc11` corresponds to block number 19517842 on the Ethereum main network. Its contents can be inspected via `https://etherscan.io/block/0x34e89f72d1bc09af19759a94c347d769a94bb0ca9d257b2efe8ad5ac0502cc11` or `https://etherscan.io/block/19517842`. This block contains 185 individual transactions, including the one where I have swapped some tokens.

Block height or **block number** is the number of blocks preceding it. Older blocks have lower height. The bigger this number, the longer the blockchain has been in existence.

Block time is the average time between blocks on the blockchain, for example, 15 seconds. The value itself is arbitrary and is chosen by the creators of the blockchain. The important thing is that this value is approximately a constant. In Bitcoin, the expected block time is 10 minutes, while in Ethereum the design calls for a block time of

15 seconds which in practice is usually between 10 and 19 seconds. Initial implementations of Creditcoin used 60 seconds, with later versions dropping down to 15 seconds. Constant block time is better for layered applications because they can know in what timeframe an operation can be considered as failed or timed out.

Genesis block is block number zero. It is usually hard-coded into the blockchain implementation, for example, your client program, and may contain metadata that's necessary for the subsequent correct operation of the chain. For example, which account is the root account or a starting balance for a set of predefined addresses.

A **fork** happens when the blockchain diverges into two potential paths forward. A fork may occur when multiple nodes produce a block at nearly the same time. The fork is resolved when subsequent blocks are added and one of the paths becomes longer than the alternatives.

A **hard fork** usually occurs when there is a change to the blockchain protocol that is not backward-compatible and requires all users to upgrade their software in order to continue participating in the network. In a hard fork scenario, the blockchain network exists as two separate digital entities which are not connected to one another. Hard forks may also occur due to bugs in the software implementation and/or networking issues which lead to consensus failure. In other words, participants in the blockchain cannot come to an agreement which path forward should be the canonical one and the result is two separate chains.

Since a blockchain is meant to be used as a source of truth, situations in which forks cannot be resolved and the result is a hard-fork are considered to be critical and potentially damaging to the blockchain.

Block **finality** refers to the irreversible confirmation of transactions on a blockchain, ensuring security and preventing double-spending. Block **finalization** is the process used to handle forks and choose the canonical chain where the majority of the participants are in agreement that blocks should not be reverted. In practical terms, this means two-thirds of all participants.

State is the current snapshot on the blockchain – the total number of blocks, the transactions recorded in these blocks, and the value(s) recorded in local storage. In practice when we talk about state, we mean a snapshot of the blockchain at a specific block. For example, the state at block 1000 is that Alice has a balance of 300 and Bob has none! In block 1010, the internal state may have changed due to transactions which have been executed in the meantime. For example, Bob has now earned a certain amount of tokens. When we talk about state, quite often we mean the last values recorded onto the blockchain ignoring previous history. Depending on the context, we may want to take into account prior history, probably if you are trying to reproduce a specific bug.

Mining is the process of performing computationally intensive cryptographic operations as part of the proof-of-work consensus algorithm. Computers performing this work are known as **miners**. Broadly speaking, we also use the term **miner** when referring to a computer and/or a human participating in a blockchain, regardless of the actual consensus algorithm.

In cryptography and blockchain, a **nonce** is an arbitrary number that can be used just once in communication. It is often a pseudo-random number issued in an authentication protocol to ensure that old communications cannot be reused in replay attacks.

Validator is a computer node which verifies the legitimacy of blockchain transactions. Also known as **miners** or **minters**, validators are the ones who participate in the consensus algorithm and gain privilege to append blocks to the blockchain. The process is entirely automated, and when we say "user," we generally mean a computer participating in the blockchain network; however, keep in mind that this computer is operated by a human, aka the user, which may also be referred to with the same terms. The term validator is commonly used with proof-of-stake consensus algorithms, while miner is more commonly used with the proof-of-work algorithm.