MOCHIMO
Post-Quantum Currency

Abstract

AN INNOVATIVE APPROACH TO CRYPTOCURRENCY that introduces a quantum-proof transaction network, true decentralization, ultra-fast transaction processing, unprecedented convergence time and unparalleled scalability—all on a trustless, peer-to-peer network running on consumer hardware.

Matt Zweil
mzweil@mochimo.org

TABLE OF CONTENTS

1 FOREWORD & ACKNOWLEDGEMENTS

WHEN THE MOCHIMO DEV TEAM BEGAN OUR WORK in the summer of 2017, we agreed to refrain from publishing a white paper until the system had a working product. Today we not only have a currency, $MCM, but a licensable, robust suite of algorithms that allow anyone to build a next generation currency atop a resilient protocol that ensures longevity, immutability and decentralization.

Contained herein is a description of the Mochimo Cryptocurrency Engine **as it is today**. These are not future plans but actual documentation of how we have already resolved many of the crucial issues plaguing cryptocurrencies today. We have implemented a series of innovative methodologies, AI and disruptive algorithms that we collectively term the **Mochimo Protocol**.

The MCM project owes a great debt of gratitude to Dr. Andreas Hülsing and Daniel Bernstein for their pioneering work in the area of post-quantum cryptography. To Hülsing, specifically, we owe thanks for vetting our cryptographic code. Additional thanks and recognition go to Albert-Laszlo Barabasi for his contributions to the field of Random Networks. His work has also influenced the underlying mechanics of our contention-resolution and distribution network. Finally, to Satoshi Nakamoto, the father of Bitcoin, whomever and wherever you are: we thank you for your vision of a decentralized currency. We are proud to be carrying your work forward to create a currency that is truly scalable and sustainable.

2 EXECUTIVE SUMMARY

MOCHIMO [$MCM] IS A THIRD-GENERATION CRYPTOCURRENCY and transaction network architected from the ground up to avoid known issues and deficiencies in existing blockchain systems. Mochimo was coded from a blank page to combine best-in-class features into one crypto ecosystem that is also future proofed through post-quantum cryptographic algorithms. As part of this protocol, the currency employs a randomized peer-to-peer network, a new consensus algorithm, and a unique proof-of-work mining technology. These pieces work together to produce a trustless, distributed ledger.

Most importantly, the various algorithms developed for the Mochimo currency include multiple innovations and features that provide an already working solution to some of the most critical problems plaguing existing and newer blockchains today.

Here is a short list of some of our innovations:

1. **ChainCrunch™ Technology**
   Proprietary technology that reduces the total size of the blockchain ensuring its ability to scale and process large number of transactions (scaling from existing 1K TPS to 20K TPS within 6.75 years); short-term and long-term scaling is not a problem for Mochimo.
2. **Trigg's Algorithm**
   Proprietary proof-of-work algorithm that ensures FIFO transaction processing with a fixed transaction fee. Mining will stay viable for all levels of miners indefinitely.
3. **The Mochimo Consensus Mechanism**
   A new system built on top of the Random Networks model that allows for high-speed convergence, orphaned chain pruning, and a mathematically provable consensus that is superior to the consensus-by-rumor model of many cryptocurrencies.
4. **Quantum Resistant Security**
   By deploying WOTS+, vetted by the EU funded PQCRYPTO research organization, to secure Mochimo addresses, and by basing the entire MCM Protocol upon quantum secure algorithms, the Mochimo development team has resolved a crucial issue that will eventually cause ECDSA-based protocols like Bitcoin, Ethereum and all ERC-20 tokens to become functionally insecure and inoperable as either a transaction network or store-of-value.

5. **Fair Distribution Model**
   Minimal premine for the dev team, absence of an ICO, self-regulating and constant mining difficulty, and the insurance of a slowly decreasing block rewards are all built in deterrents to ensure a fair distribution of MCM and maintain accessibility to "late" adopters.

The Mochimo development team is led by system architect Matt Zweil, an expert network architect who has designed and deployed some of the most ambitious projects in the areas of Transaction Networking, Datacenter Design, and Service Provider Networking in the industry. Mochimo's lead developer is Trigg, a master C programmer and an AI researcher who has been developing innovative systems since the late 70's. Together they have created the MCM Protocol and the ChainCrunch™Technology. With the support of the broader Mochimo development team, Matt and Trigg have achieved a working protocol that is the substance of this paper.

3 MOCHIMO PROTOCOL DESIGN PRINCIPLES

AFTER INVESTING AND CONTRIBUTING to the first and second wave of cryptocurrencies since 2009, a team of blockchain veterans came together in early 2017 to start the Mochimo project. As cryptocurrency purists, one of their first tasks was to codify the set of principles that would govern the design.

The starting goals of MCM were a future-proof cryptocurrency that would be truly decentralized in nature, trustless, immutable, and scalable without upper bounds. We have achieved all four tenets and much more with our protocol.

In summary, the Mochimo Cryptocurrency Network is a Peer-to-Peer, trustless distributed ledger with high-speed convergence and strong double-spend protections. Mochimo is not a fork nor could it achieve what was done simply by re-engineering existing code. Instead, the Mochimo crypto ecosystem is a complete reimplementation of a blockchain distributed ledger based on Satoshi Nakamoto's original vision but enhanced by the benefit of years of experiential updates. Of these tenets, the most crucial one was decentralization.

Today, cryptocurrencies fall in two basic categories:

1. **Truly decentralized and therefore trustless**
2. **Semi-centralized**

The semi-centralized currencies should be rejected without exception. Centralization of anything—exchanges, credit information, currency—invites attack. Instead of an independent ledger—the very essence of the blockchain—you get a centralized authority, right or wrong, that dictates to the network at large what the current state of the ledger is.

Of course, some people might call that flaw a feature. Centralized systems will often distract people from the inherent flaws in their architecture by highlighting extraordinary transaction throughput numbers. They fail to mention that the cost is the evisceration of the trustless environment. In fact, exorbitant transaction per second ("TPS") advertisements are often the initial clue that a system's creators intend to retain control of your assets.

3 MOCHIMO PROTOCOL DESIGN PRINCIPLES

How does 'fast speed' translate to 'control'? A centralized authority can process hundreds of thousands (if not millions) of TPS not because their consensus mechanism is so efficient but because all consensus is bypassed. To use a political analogy, dictatorships are efficient but the government is no longer a legitimate representative of the people nor does it serve the people. The obstacle to achieving scalability in cryptocurrencies today is that the speed of current consensus mechanisms is the bottleneck.

So how do we maintain or increase speed without handing over control to a central authority? As we see it, the primary design challenges are as follows:

- Ensure that the blockchain size does not grow out of control Maintain the bandwidth requirements for communication as accessible to the average person. Anyone should be able to simply and easily stand up a node and join the network, fully synchronize the ledger, and begin to process transactions and mine blocks.
- Allow for rapid dissemination of transactions, block updates, and rapid convergence, all enabled by effective and mathematically provable contention resolution.

To solve these issues, the Mochimo ecosystem introduces several innovations, chief among these being "ChainCrunch™" which allows any individual node to maintain a complete view of the ledger while discarding old blocks. Mochimo also features exceptionally fast convergence and orphaned chain pruning. Lastly, in keeping with this vision, Mochimo has some of the most well-documented code in the crypto world.

3.1 AUTONOMOUS DECENTRALIZATION

Our first tenet is that a cryptocurrency, to be truly decentralized, can have no actors, either the miners or developers, able to dictate policy direction after its release. Therefore, the code will be the whole of the law governing the system, and no one should control the code.

It is for this reason that Mochimo rejects all attempts by recent cryptocurrencies to introduce trusted nodes, voting mechanisms, proof-of-stake algorithms, or delegated proof-of-stake algorithms. Furthermore, we reject without reservation any consolidation of mining power as it allows those actors to dictate policy through brute force. Any mechanism introduced into crypto that allows a given operator to exert influence greater than any other actor can and will eventually allow those in positions of power to further aggregate that power. This will eventually disrupt the autonomy of the network. This manipulation has already happened across almost every existing cryptocurrency and can collectively be referred to as the tendency toward centralization. It also spells the long-term death of these ledgers.

3.2 THE SELF-HEALING LEDGER

The Mochimo team believes that each individual network node must be able to determine the state of the network, the ledger, and any given transaction without the need to petition an authoritative source.

Trusting a single authority is the weak link of any autonomous system.

For this reason, the Mochimo design rejects the concept of “Master Nodes”, “Super Nodes”, “Trusted Nodes” and all other euphemisms designed to give a centralized authority the ability to regulate the behavior of the network. Put more succinctly: if the network requires parties to gain trust between each other in any sense to operate, then that ledger is mutable.

IF THE LEDGER IS MUTABLE IN ANY WAY, THE CURRENCY IS WORTHLESS.

Instead, when contention arises within the MCM network, the nodes themselves will mathematically determine the master chain and autonomously prune any orphaned chains. Likewise, transactions and block solutions are not relayed through central nodes, but rather peer propagated via multicast mechanisms to the whole network.

At every level, the Mochimo protocol rejects the centralization of processing, transaction distribution, peer detection, contention resolution, voting, tie-breaking, and code version enforcement. All of these mechanisms are weaknesses. They create opportunities to make the ledger mutable and any one of them can be abused by a group to change the outcomes and state of the ledger, rendering it mutable. It is only through running a trustless system, i.e. where every node has all the information it needs to make decisions and arrive at consensus, that end users of the currency can truly trust the transaction network.

3.3 EGALITARIAN MINING INCLUSION

Scaling is the largest issue facing first and second-generation cryptocurrencies today. Most of today's top 100 currencies have computing, bandwidth and storage requirements that are growing at a greater rate than is sustainable on the hardware available to the average person. As mining difficulty rises, the average person can no longer mine.

Once mining is effectively restricted to the larger organizations that have sufficient resources to stand up high-powered computing and storage resources, centralized control of the coin begins. As mining is centralized, these miners also centralize the nodes. Through this, they begin to control the direction of the coin's development, delaying or preventing any innovation that decreases their mining interest. Shortly after this, hostile forking and other detrimental behavior ensues.

Similarly, no one wants to enter a situation in which they will always be powerless and, if the decisions are always against them, they will eventually exit the currency in favor of a more egalitarian situation.

As the centralized majority fails to keep the newer and smaller players, the currency moves away from the idea of creating a sustainable store of value that can be used daily by the entire population towards a currency that reflects only the interests of whomever is mining it the most.

Users will always be able to mine Mochimo without being marginalized by bad actors.

4 MOCHIMO DESIGN SOLVES MANY ISSUES

AS CRYPTOCURRENCY HAS MATURED, various problems have emerged as a result of large scale adoption. Many of the second generation coins are attempts to solve one or more of these issues. However, unlike those piecemeal protocols, the Mochimo team developed a holistic and anticipatory solution by incorporating a series of cryptocurrency design innovations that have solved all the following problems (which will be addressed in detail below):

- The Threat of Quantum Computers
- A Long-Term Solution for Network Scalability
- Ensuring FIFO Transactions and No Transaction Queues
- Transaction Throughput and Security

4.1 THE THREAT OF QUANTUM COMPUTERS

The first and most notable problem that Mochimo solves is the threat to the cryptocurrency ecosystem posed by the rapidly approaching rise of quantum computers. Currently, the majority of blockchain systems and cryptocurrencies have their addresses and balances protected by quantum-insecure Digital Signature Algorithms. The most commonly deployed of these is ECDSA (used by Bitcoin, Ethereum, and all ERC-20 tokens), and it will be paper-thin in the face of a quantum computing attack.

Breaking this Elliptical Curve Digital Signature Algorithm with conventional computers is computationally intractable. Yet for QCs, it is one of the first developmental goals for this class of computer. In light of this, these ECDSA cryptocurrencies are, while a remarkable achievement, dangerously exposed1. They cannot, therefore, realistically be viewed as a long-term store of value. Nor is there a magic bandage that can be added on to the underlying code of these cryptocurrencies to make them quantum resistant.

This is why quantum resistance must be built from the ground up and why we didn't borrow from, fork or base our design upon any existing protocol. Organizations and individuals aware of the quantum shortcomings of ECDSA have begun to codify new cryptographic protocol standards that are considered "Post-Quantum Secure". The most notable of these organizations is the PQCRYPTO working group funded by the European Union. Their "ICT-645622" document is a reference standard for quantum secure encryption algorithms based on the current and projected capabilities of Quantum Computers.

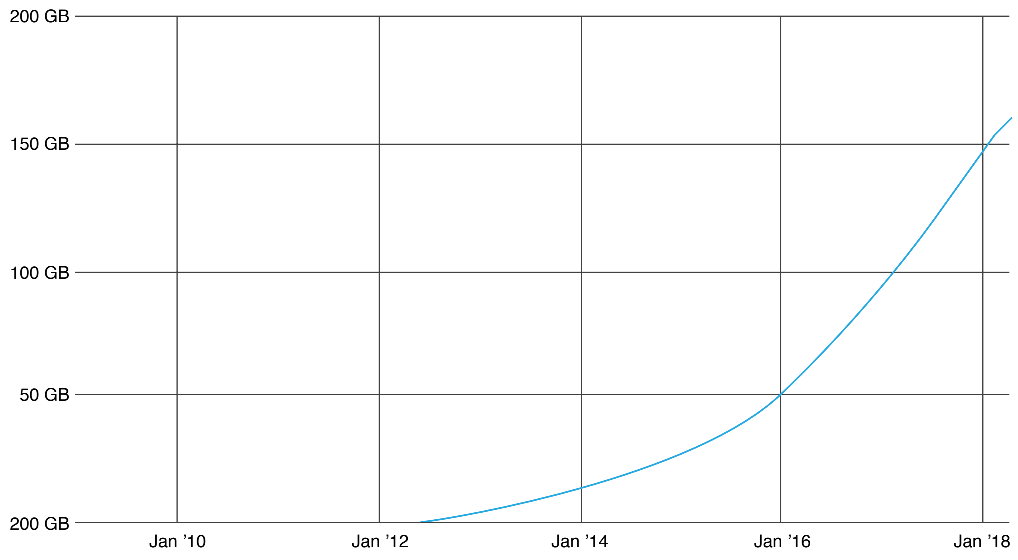
Within that standard they've recommended a handful of Digital Signature Algorithms that will remain computationally intractable irrespective of the improvements seen in Quantum Computing capability2. From this list, the Mochimo Development team selected the XMSS+ with the WOTS+ variant of the Winternitz One-Time Signature proposed and proven by Andreas Hülsing in September 2017.

THE MOCHIMO DEV CONTRACTED WITH HÜLSING in February of 2018, to perform a complete review of our cryptographic code. The Mochimo Project's use of the WOTS+ Digital Signature Algorithm and our ANSI-C (1989) implementation of this quantum-proof algorithm for our addresses and Protocol has now been thoroughly reviewed by Andreas Hülsing himself, the creator of the algorithm. Our implementation was found to be free from material defect and Hülsing's final code review will be published on the Mochimo website alongside this whitepaper.

4.2 LONG TERM SOLUTION TO NETWORK SCALABILITY

Mochimo is a transaction network; as such, scalability is a primary consideration. Many protocols have irresponsibly allowed their blockchain size to grow indefinitely, routinely getting into the hundreds of Gigabytes and even Terabytes.

Bitcoin (BTC) Blockchain Storage Requirement Growth in Gigabytes



4 MOCHIMO DESIGN SOLVES MANY ISSUES

In the case of Bitcoin4, we can clearly see that the blockchain size growth doubles the size of the blockchain every 12-16 months. The size of the Bitcoin blockchain, which has a comparatively small address size and signature size, has grown along an exponential curve and is now sitting at over 180GB of raw data as of March of 2018. Ethereum, famous for its frequent network congestion, has had shutdowns caused by a growth that has been far more aggressive, with its full blockchain size (including traces) now exceeding 1TB of requisite storage space.

Other currencies will fare no better as network demands and blockchain size increases. In light of this congestion, it's easy to see why there might be a resistance to implementing quantum security measures. The size of the addresses and signatures are an order of magnitude larger than those employed by protocols such as Bitcoin or Ethereum.

However, despite the congestion, these developers invariably claim that Moore's predictions on the rate of future technology will eventually catch up to their out-of-control blockchain size growth. This appeal to "Moore's Law" is fundamentally flawed as cryptocurrency critics have empirically observed a growth in storage consumption across virtually ALL blockchains that exceeds the technological growth target of 18 months as predicted by Moore's Law.

Mochimo, however, has already solved the issue of large and runaway blockchain growth using an innovative blockchain processing algorithm called ChainCrunch™. ChainCrunch™ is a proprietary Mochimo tech that allows a user to operate a full node, but maintain only a small percentage of the historical blockchain data. ChainCrunch™ is secured by HASH256 and is quantum proof. Because of this innovation—discussed in greater detail later in this paper—the

MOCHIMO'S RADICAL CHAINCRUNCH™ SOLUTION to scaling doesn't stop there. It also allows the system to conserve storage and radically improve data lookup speeds. With ChainCrunch™, new nodes can join the network and fully synchronize within minutes, rather than days or weeks. This technology allows for unprecedented levels of scaling without requiring large amounts of storage space. Additionally, the ChainCrunch™is only one component of a suite of innovations in the MCM Protocol that allow for near-instantaneous lookup of address balances and transaction data.

size of the Mochimo blockchain will not grow. Instead it will remain exquisitely and reliably small no matter how many years the network is operating or how many transactions we process.

4.3 REINVENTING TRANSACTION FEES TO ENSURE F.I.F.O. PROCESSING

The problem with existing transaction fee systems is that they take what should be an egalitarian, decentralized network, and break it by incentivizing miners to process transactions in the order of which person can afford to pay the most. When miners select for the highest fee, they leave the low or no-fee transactions to wait for hours in the purgatory mempool of busy networks. In the case of Ethereum ICOs, people have been paying absurd amounts to "jump the line." This is not a behavior that promotes a reliable, scalable system. Only changing the mining incentives can stop this sort of behavior.

The Mochimo dev team believes that incentivizing miners to prefer one transaction over another is counterproductive to a healthy network. For that reason, the Mochimo protocol takes a novel fixed-fee approach to transaction processing. The cost to send a transaction on the Mochimo network is now and will always be: 0.000005 $MCM. To give you an idea of how small this cost is, if the Mochimo marketcap grows to overtake the entire altcoin market cap and one

SO HOW ARE MINERS INCENTIVIZED NOW? The small, fixed, transaction fee for Mochimo encourages miners to stack as many transactions into their candidate block as possible, helping ensure FIFO transactions and preventing MEMPOOL queuing as seen in other cryptocurrency networks.

Mochimo coin were worth $25,000 USD, the transaction cost would be less than $0.13 USD. Therefore, it is safe to describe the transaction fee as trivially small for quite a few years. This will encourage day-to-day use of the currency.

4.4 TRANSACTION THROUGHPUT AND SECURITY

The Mochimo Protocol requires every transaction to have the following 6 basic elements: Source address, Destination address, Change Address, Amount sent, Mining fee (fixed), and Balance Change amount.

It also has the following important restriction: the source address is always emptied and destroyed upon use. The system checks that the amount sent plus the balance change amount As a result, the size of every transaction in bytes is fixed, the inputs and outputs are trivially simple, and the protection for lost coins is straightforward.

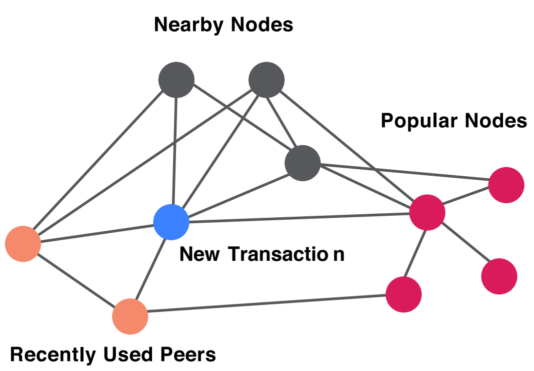
Furthermore, you cannot send a transaction without making a full accounting of all the currency in the existing source address. Nor can you aggregate multiple inputs and outputs. All wallet and balance management for addresses is a client-side function of the wallet software.

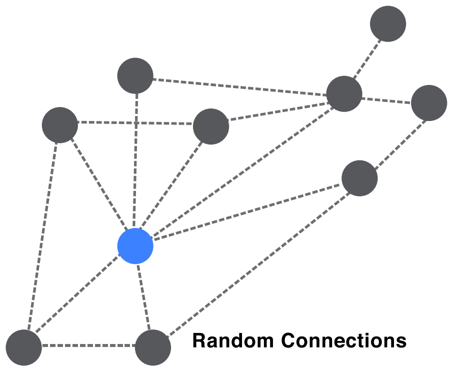
When combined with the extraordinary speed increase enabled by Mochimo's ChainCrunch™ technology, the lookup, validation, and execution speed for transactions on our network are among the fastest in the industry. More importantly, with ChainCrunch™, these speeds do not slow down or stagnate as the network grows in size or transaction throughput, but instead remain consistently fast and will scale above and beyond the initial capacity as the average node's processing hardware increases. So Mochimo is fast and will only get faster with time.

5 TECHNICAL OVERVIEW OF KEY FEATURES

5.1 CONNECTION MANAGEMENT USING THE RANDOM NETWORK MODEL

The Average Blockchain Non-Random Dissemination of Information Across the Blockchain Network	**Mochimo Blockchain** Non-Random Dissemination of Information Across the Blockchain Network





The average blockchain picks only nearby, popular, or recently used peers (nodes). Repetitively picking from a much smaller group of similar nodes allows various ways to attack the network.

Every node on the Mochimo network is opening and closing random connections across the entire fabric of the network. The server randomly populates the 16-member current peer list with second-degree connections, and attempts to establish connections with each, always pruning failed connections. For each failed connection, the server issues a GET_PEER to a successful second-degree connection, shuffles the response, and continues to populate its current peer list with unique third-degree connections.

RATHER THAN CONNECTING ONLY TO RECENT, WELL-KNOWN, OR GEOGRAPHICALLY CLOSE PEERS, the design goal of MCM connection management was to ensure a trustless environment whereby every node on the Mochimo network is opening and closing random connections across the entire fabric of the network. The Random Networks protocol is designed to disseminate all received transactions to the rest of the network quickly and asynchronously. This random network model was formally analyzed by Albert-Laszlo Barabasi in his Network Science publication: Random Networks5.

The process: the server distribution is seeded with a shortlist of well-known peers. These are just the standard nodes that are included with the distribution. This list may be overridden by the -S switch at the command line, with the user supplying either a specific peer to connect to initially, or a file containing a list of peers to connect to. While the Mochimo Dev team set up those well-known nodes, they are no different than other nodes.

On initialization, the server loads the well-known peer list or the user-provided list, shuffles them, and picks one at random to connect to. On connection, the server issues OP_CODE: GET_PEER, and if successful, receives a copy of a recent peer list from that first-degree connection. From that list, the server randomly populates the 16-member current peer list with second-degree connections, and attempts to establish connections with each, always pruning failed connections. For each failed connection, the server issues a GET_PEER to a successful second-degree connection, shuffles the response, and continues to populate its current peer list with unique third-degree connections.

A ROTATING, RANDOMIZED NETWORK MEANS that the degree of separation between any two nodes on the open internet is 16^D = N, where D is the number of degrees of separation and N is the number nodes on the network. For example, in a network with exactly 4096 nodes, the average degrees of separation between any two nodes will be 16^3 = 4096, or 3 hops. For a network with 65536 nodes, the average increases to 4 hops.

The process of populating the peer list repeats until the current peer list contains 16-active peers of at least second and third-degree separation from the initial well-known peer. As current peers age out or become unreachable, fourth degree connections are rotated into the current peer list the same way i.e., by extracting a peer list from the last peer added. In this way, each Mochimo server is constantly rotating through new and diverse peers throughout the world.

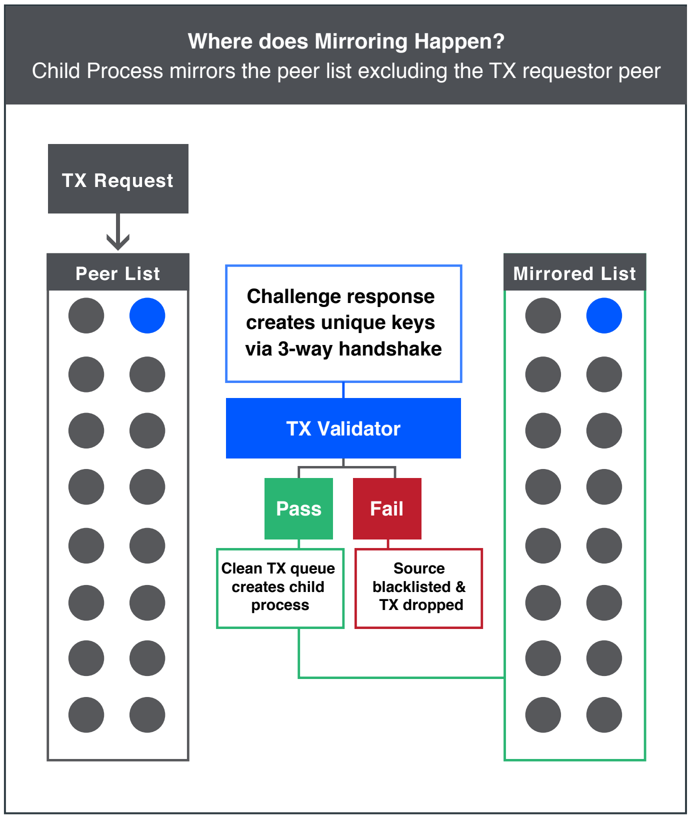
Among other things, this Random Networks model can be used to mathematically determine the average number of degrees of separation between any two nodes on the Mochimo Network. By design, we are randomly crawling the network for new nodes, and each Mochimo server works to dynamically ensure the rapid dissemination of transaction mirroring and block distribution.

How does it scale? With this network type, if the network size in nodes N is equivalent to the current approximate size of Bitcoin, the average number of degrees of separation between any two nodes on the Mochimo network, using a rotating, random selection of k = 16, will be between 3 and 4. This means transaction dissemination, block found messaging, and network-wide convergence happens in at most 3-4 hops for the relevant data6.

5.2 TRANSACTION MIRRORING

The design goal of transaction mirroring is to leverage the Random Networks model.

To understand where mirroring fits in, first consider that the primary server process maintains a rotating list of current peers, receives transactions requests, queues them, validates their signatures, and then spawns child processes to perform transaction mirroring.


Where does Mirroring Happen?
Child Process mirrors the peer list excluding the TX requestor peer
TX Request
Peer List
Challenge response creates unique keys via 3-way handshake
TX Validator
Pass
Fail
Clean TX queue creates child process
Source blacklisted & TX dropped
Mirrored List

The process: the server maintains a socket-table of peer connections to the 16 most recent peers, aka the **CURRENT_PEER table**, and polls each for inbound requests. The server completes a challenge-response three-way handshake to establish unique communication keys for each new inbound request.

A transaction is received from a peer with OP_CODE: OP_TX. The server parses the transaction and passes it to the Transaction Validator function (TX_VAL).

The **TX_VAL** function validates a number of parameters, including the following: Is the peer advertising the same current block as the local system? Are transaction parameters valid against the local ledger? It then performs duplicate detection; if all transaction parameters are good, it performs a signature validation.

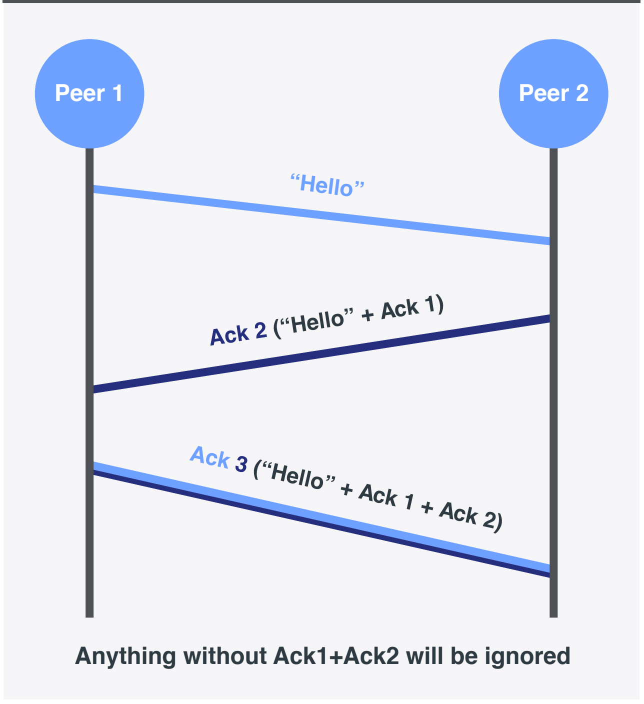
If valid, the TX is moved into the **CLEAN_TX_QUEUE**. If the TX is bad, the originator is blacklisted and the TX is dropped. For clean transactions, the server asynchronously spawns a child process to service the CLEAN_TX_QUEUE. The child process services the queue by mirroring transactions to up to 16 peers in the CURRENT_PEER_LIST, excluding the peer that originated the transaction, and then exits. In the meantime, the server continues its normal processing loop. TX Mirroring uses an inbound source-hash validation to ensure TXs are not mirrored back to their originators.

5.3 THREE-WAY HANDSHAKE

In all peer-to-peer communication, Mochimo deploys a security feature to prevent denial of service attacks, spoofing and man-in-the-middle attacks. This three-way handshake is found in many network protocols; here is how it works in Mochimo.

The process: for each new outbound connection, the Mochimo server generates a random 16-bit identifier in a Hello message. This identifier is referred to as ID1. The receiving peer responds with a Hello-ack containing ID1 and its own unique, randomly generated identifier, ID2. The original peer completes the three-way handshake with an ACK-ACK containing both ID1 & ID2, and from that point forward the peers are said to be "Fully Adjacent". They may now send and receive within that one session by tagging the traffic accordingly.

The Three Way Handshake
Anything without Ack1+Ack2 will be ignored


Peer 1
Peer 2
"Hello"
Ack 2 ("Hello" + Ack 1)
Ack 3 ("Hello" + Ack 1 + Ack 2)
Anything without Ack1+Ack2 will be ignored

These identifiers are unique to that short-lived session, preventing denial-of-service that could occur through node impersonation and man-in-the-middle attacks. Because ID1 & ID2 are renegotiated with each new transaction, the Mochimo Three-Way Handshake provides fast, simple and disposable security.

Any communications received from peers that are:

1. Not part of the Three-Way Handshake state machine, or **2.** Are not signed with the correct IDs are simply ignored.

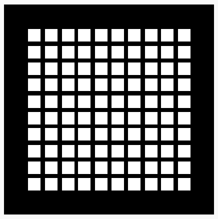
5.4 RAPID SERVER INITIALIZATION

When the Mochimo server comes online, it will be in one of two states: Clean Boot, or Graceful Restart. The system will only be in Graceful Restart if the system was shut down from the monitor by intervention from the end user. For all other cases, when the system goes down (for example to resolve contention, prune an orphaned chain, or due to a fatal error) all state information for the server is washed. This includes the peer lists, the local ledger, the full blockchain, any candidate blocks on the disk, etc.

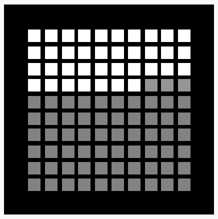
The process: the Mochimo system maintains stability by returning to the Clean Boot state and leverages the fast convergence feature of Mochimo to recreate its blockchain / local ledger any time corruption or contention is

Clean Boot Process
Part 1: Everything is wiped clean and two Aeons worth of blocks are rebuilt from the Trailer File

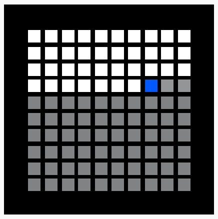
1 **Retrieve former Genesis Aeon and rebuild on the node that was wiped blank.**



2 **Rebuild current Aeon back to current block.**



3 **Wait for a new block to be formed by a peer outside the quorum.**



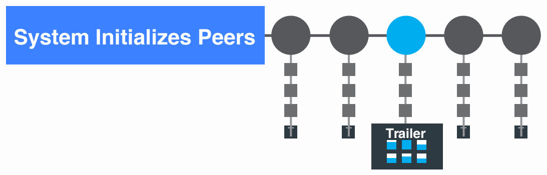
confirmed. Because contention is exceptionally rare in the Mochimo implementation, a software-forced Clean Boot will not occur often.

On Clean Boot, the system initializes its core IP list and creates a list of random peers of at minimum second and third degree.

The server then makes use of our MCM proprietary Quorum function to identify the longest blockchain among the peers present in its peer list, aka the "master chain". The longest blockchain is the chain with the most work performed. To identify that, Mochimo nodes maintain a historic block Trailer File with chain information going back to Block 0, the original Genesis Block. Contained within that file are 100-byte entries for every block ever solved that form a verifiably linked list of block solutions, including nonce and hash, and the time signatures used to derive the difficulty requirement for each block.

Master Chain Discovery
Part 2: The system compares the Trailer Files to find the chain with the blocks that did the most work

Nodes have Trailer Files with Historical Data



The server uses the MCM Quorum function to enter the Trailer File to "weigh" the difficulty of each block. The chain with the heaviest blocks counted in the trail wins the discovery process.

The file is walked from beginning to end; for each solved block, the difficulty of solving the block is aggregated into a counter. The weight given to a solved block in the T-File chain is computed as 2^(Difficulty - 1), meaning a block solved with Difficulty 35 has twice the weight of one solved at Difficulty 34. Because the T-File chain is provably linked, and contains solved blocks of X difficulty, by requesting a peer's T-File, we can compute the total amount of work that has been performed on that peer's blockchain from the original genesis block to the most recent block. If the total weight of that chain is the highest visible in the network, that node is on the master chain and can be provisionally trusted.

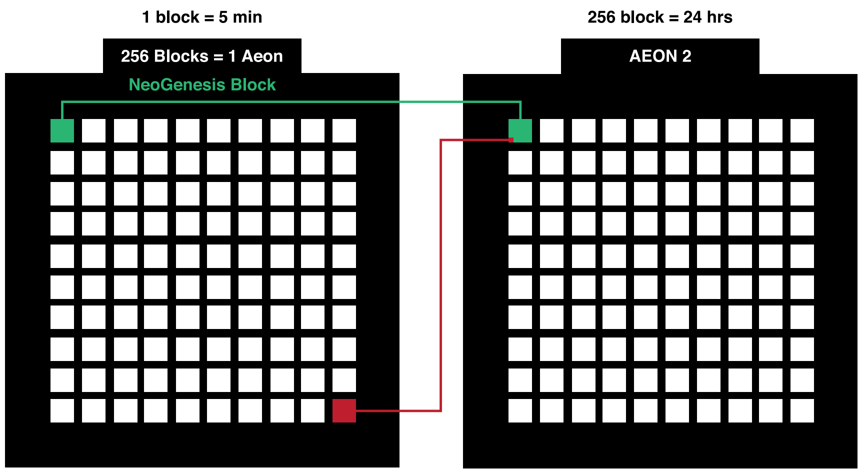
5.5 THE TECH BEHIND CHAINCRUNCH™

ONE OF MOCHIMO'S MOST IMPORTANT PROPRIETARY INNOVATIONS, ChainCrunch™, allows the network to maintain a complete picture of the ledger, validate blocks with perfect fidelity, and rebuild the local database of a new or existing peer in a few short minutes instead of weeks. Additionally, ChainCrunch™ allows for extremely rapid lookup for transaction validation while maintaining a small block chain size and storage requirement that is just a fraction of what is required for all other cryptocurrency networks.

The process: a Mochimo "Aeon" is 256 blocks, with block solve times spaced at 337.5 seconds apart, resulting in an average Aeon lifetime of 86,400 seconds, or one day. The Mochimo server maintains a local database on disk that is a sorted list, indexed by address, of every address with a balance in the network. To put this in context, the system makes additions and deltas to that database based on blocks 1 through 255 by processing valid transactions and sorting the changes with each successful transaction. Having arrived at Block 255, the system writes the database to the disk as block 256, hashes it, and appends a trailer. This special block is referred to as a "neogenesis block". From the neogenesis block, the system continues forward by attempting to solve block 257.

From the moment this neogenesis block is written to the disk, all historical data from the Blockchain is no longer needed for the system to function.

Block & Aeon Details



Database is written to the last block, hashed, and appended to a Trailer File.

511 is the maximum # of blocks held at any given time.

THE NEOGENESIS BLOCK allows the system to discard almost all blocks in the chain, but still validate transactions with perfect fidelity. With ChainCrunch™ and the neogenesis block, the system will never have to store more than 512 blocks on the disk as restarting a new system requires rebuilding from the first previous neogenesis block, i.e., not the neogenesis block of the current Aeon.

When a new system comes online or restarts as a result of failing a contention check, the system will use the random networks model to find a quorum of peers that are on the master chain. From that point, the Mochimo server issues a request to one random member of that quorum and pulls down the first previous neogenesis block.

SINCE A NEW NEOGENESIS BLOCK IS PRODUCED EVERY 256 BLOCKS, the requested block will be the Current Block on the network minus (at most) 511 blocks. As such, the number of blocks required to fully sync a Mochimo node will always remain between 257 and 511. As you can see in the diagram on page 16, no matter where you are in the second Aeon, you're always less than 512 blocks away from that first previous neogenesis block.

The untrusted neogenesis block that the server downloads contains a raw ledger file, listing the balances of every address with a balance in the Mochimo network. The Mochimo server imports this ledger in an untrusted state and uses it to validate the ensuing 256 blocks on the chain, pulling them from the Quorum swarm one at a time, and validating them against the ledger, making adjustments as necessary, and building on the local Trailer File.

On arriving at the 256th block, the server generates its own neogenesis block, which exists in a provisionally trusted state, and begins to validate all existing blocks in the current Aeon, until it arrives at the same Current Block as the Quorum, validating the block hash against theirs. From that point, the server waits for a new block to be solved by a non-Quorum member, and if that block can be validated against the Server's local state, the Server is said to be "synchronized", and local state information is now trusted.

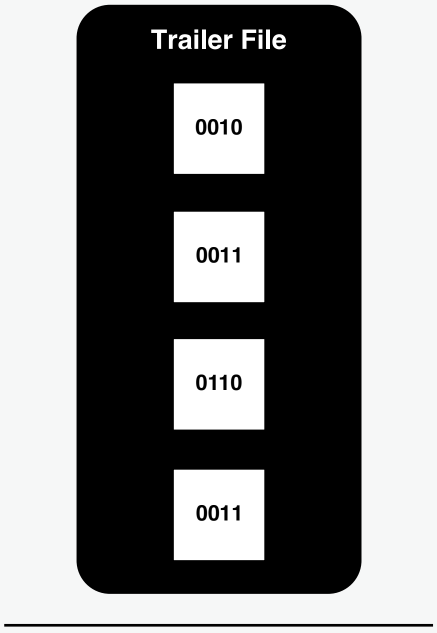
The server now moves into the "Online" state and sends and receives transactions, attempts to solve blocks, etc. as usual. This entire process, from beginning to end, will typically take a few minutes depending on how long it takes for the next block to be solved.

5.6 CONSENSUS ALGORITHM

A consensus algorithm answers the question: "How does a node know whether it is on the master chain or an orphaned chain?" Mochimo answers this by validating "BLOCK FOUND" messages according to the amount of work performed by the blockchain that the advertising peer is on. To arrive at that number, we must first introduce the idea of the Trailer File and Chain Weight.

What is a Block weight and what does it do?
The total weight of the block determines which chain becomes the master chain.

We calculate the amount of work done on each block in binary and add it up.



Get the total weight

The process: the Mochimo Trailer File is a linked list of every block trailer since the original Block 0 genesis block, no matter what Aeon the system is on or how many thousands, or millions, of blocks have passed. Each trailer is 100 bytes of data, so even if Mochimo runs for thousands of years, it won't be prohibitive to store this amount of data. The Trailer File is immutable, it includes: the start and solve times of each block, the block hash, the nonce/hash that solved it and the difficulty requirement that was present at the time of the solution. This linked list of trailers is used to compute the aggregate work of the chain that claims to have found the block. That value is referred to as the **"weight" of the chain**.

THE TRAILER FILE is part of the underlying mechanic of the ChainCrunch™ that allows us to discard block data but retain a chained list of block solutions from the first block until present.

To determine the work performed, we simply add the difficulty values of every block together in binary. A block that was solved with a difficulty of 34, for example, requires a hash output with 34 leading zeros. When we compute the weight, we add 2^34 to the total weight of the chain on which that block is solved. If the next block is solved with a difficulty of 35, the weight of that block is twice what the previous was.

As the network increases in size and number of miners, the difficulty increases, making each solved block, over time, heavier than previous blocks. Importantly, as we validate a T-File, we validate each hash and nonce, the start and solve times, recreate the Difficulty (which is computed from block 0), and then append to the Weight for the candidate chain.

Using chain weight, as opposed to blockchain length in terms of block numbers, allows us to quickly and deterministically prove that one chain was created with more total work than another. It also prevents malicious actors from trying to introduce false chains or false blocks/ledgers into the network, as, in order to falsify a block, they would have to perform the sum total of all work the network has ever performed. When a peer issues a claim of “BLOCK FOUND”, attached to that claim are four items: the hash of the block, the hash of the previous block, the nonce, and the difficulty.

On first pass, the receiving peer checks to see if the previous block hash indicated by the BLOCK FOUND message matches its current block. If it finds the message, the peer performs a sanity check on the advertised difficulty, confirming that it matches the expected difficulty. If that check fails, the advertising peer is blacklisted. Having confirmed that the peer is both on the correct chain and that the difficulty matches, the local peer requests the block from that peer.

Upon receiving the block, the peer invokes the **Block Validator**, which validates every transaction included in the block, constructs a candidate block, and passes it and the supplied nonce through the mining algorithm. If the block is valid, the local peer suspends mining, flushes its transaction queues, and invokes the **Block Updater**. The Block Updater performs all relevant global variable deltas, updates the local ledger database, and, if we are at an Aeon boundary, invokes the neogenesis routine.

5.7 CONTENTION ALGORITHM

Contention only occurs when two or more nodes in the network solve a block at roughly the same time. Due to the Random Networks model and setting the average solve time to 337.5 seconds, this is relatively rare in Mochimo but when it occurs, the consensus algorithm very rapidly converges the network in the following fashion.

The process: having received a “BLOCK FOUND” message, a node proceeds to update as indicated in the Consensus Algorithm section of this code. When the network enters contention, meaning there is more than one chain on the network of the same max length, we can expect that the node will receive yet another message indicating “BLOCK FOUND”, but for the block the node is already on. This message will also have a different hash, which signals the node that contention is present. The node ignores the BLOCK FOUND, because the block number indicated is not higher that the node’s current block, and the weight indicated is the same.

In this case, it is fair to say that there are two chains on the network, each with equivalent weight. Because block propagation can span the network in under .5 seconds, the expectation is that we will see a second chain on the network in N * (337.5/.5) blocks where N is the number of nodes on the network. However, the extent of the propagation of the second chain will be severely limited in most cases.

The way this is resolved is fairly straightforward: when the next block is mined, irrespective of how many simultaneous chains may have been mined, all nodes will receive a BLOCK FOUND message for a higher block number with a superior weight. With that message it will be apparent that the advertising peer is not on the same chain, as each advertised BLOCK FOUND also indicates the previous block’s hash. Since that previous block’s hash won’t match ours, we know that a different chain than ours is claiming to be the master chain.

At this point, to resolve the contention, the local node asks the advertising peer for its recent hashlist. The peer responds by providing the hash of every block from the neogenesis block to the present block. The local peer compares this list to its own hashlist and seeks backward to find a match. If a match is found, the node is on an orphaned chain. Some additional validation checks are performed to ensure the block isn’t being spoofed, and, upon validation, the receiving node flushes all state and reboots.

Note that the ultra-rapid convergence of the network allows us to resolve contention by allowing nodes that find themselves on orphaned chains to leave the network and re-sync. Additionally, it is possible for the chain to split not just once, but a second time. However, the likelihood of that occurring is 1 in (N * (337.5/.5)^2 blocks.

It is very important to understand that the number of active chains on the network is never greater than two, and one BLOCK FOUND message after contention occurs does not increase the number of chains on the network, but instead increases the number of nodes that are flushing state and rebooting.

6 NOTABLE CURRENCY STATISTICS

The following are some of the key Mochimo characteristics:

Maximum Supply: 76,533,882
Mineable Coins: 71,776,816 (93.8%)
Mining Algorithm: Trigg's Algorithm – PoW
Difficulty Adjustment: Every Block
Target Block Time: 337.5 Seconds
Genesis Block: June 25th, 2018
Network TX Fee: .0000005 MCM (fixed)
Starting reward: 5.0 MCM / Block
Per Block Reward Increment (through Block 373,760 4 Years: .00015 MCM
Max Reward (Block 373,760): 59.17 MCM
Per Block Reward Decrement (through Block 2,097,152 22 Years): .000028488 MCM
Final Reward (Block 2,097,152): 5 MCM
Total Mining Duration: ~22 years

Premine Details:

Total Premine: 6.34% (4.76M MCM)
Dev Team Compensation Premine: 4.18% (3.2M MCM)
Other Premine (Managed by the Mochimo Foundation): 2.16% (1.56M MCM)
Genesis Block: June 25th 2018 23:40 UTC

7 KEY REFERENCES

1. Bernstein, et al., https://eprint.iacr.org/2017/314.pdf
2. PQCRYPTO, https://pqcrypto.eu.org/docs/initial-recommendations.pdf
3. Hülsing, https://eprint.iacr.org/2017/965.pdf
4. https://blockchain.info/charts/blocks-size?timespan=all
5. Barabasi, Albert-Laszlo, http://barabasi.com/f/624.pdf
6. BITNODES, https://bitnodes.earn.com/