The Liquidum Blockchain

Michael Graczyk David Gudeman

{michael, david}@liquidum.org

Version 0.1.0
November 27, 2021

Abstract

We present Liquidum, a new blockchain that combines Ethereum's EVM application layer with Chia's consensus engine. EVM compatibility ensures that any Ethereum smart contract, and tools like Metamask work with Liquidum without modification. Chia's consensus engine makes Liquidum decentralized, scalable, and energy efficient. Liquidum was built with the goal of developing a massive, open ecosystem in the spirit of highly successful open source projects. Our mission is to grow the Liquidum ecosystem by actively enabling **value creation** across all aspects of the project. We believe that by commiting to value creating and remaining pragmaticly flexible about everything else, Liquidum will enjoy sustainable, long-term growth.

1 Introduction

Liquidum is a new blockchain that combines Ethereum's EVM application layer with Chia's consensus engine. EVM compatibility ensures that any Ethereum smart contract, and tools like Metamask work with Liquidum without modification. Chia's consensus engine makes Liquidum decentralized, scalable, and energy efficient. Liquidum was built with the goal of developing a massive, open ecosystem in the spirit of highly successful open source projects. Our mission is to grow the Liquidum ecosystem by actively enabling **value creation** across all aspects of the project. We believe that by commiting to value creating and remaining pragmaticly flexible about everything else, Liquidum will enjoy sustainable, long-term growth.

Our goal is simple. We want to build a completely open and free blockchain ecosystem, and grow it to be as large as possible. By "open", we mean that the chain is permissionless, the core code is open source, and development and decision making is done fully in public view.

We studied the history of hundreds of blockchains and found that many succeed and fail for the same reasons as other open source projects. Projects are successful when they focus on creating value for users. Other pursuits may lead to interesting technology, but are unlikely to

Our vision is to pragmatically and explicitly focus on **value creation** above all else. Some blockchains build for scalability or decentralization, some try to balance these and other traits. Liquidum intentionally commits to none of these. Instead, we see value creation as an end in itself. Examples of value creation include dApps, financial products, mining tools, funded cryptography research, or anything created in pursuit of economic interests manifested by Liquidum.

Following patterns in successful open source projects, the project will initially be lead by a small founding team. Because the team will receive a portion of premined

tokens, we are highly motived to ensure the long term success of the project. Over time, complete control will transition from the founding team to complete community management.

Since the release of the Bitcoin whitepaper [10] in 2009, the blockchain space has grown into a multi-trillion dollar industry with millions of participants. Despite the overall growth, most new blockchain projects fail within their first few years. Why do some projects grow into thriving dApp ecosystems with billions of dollars in value, while others are never used by more than a few early adopters?

We studied the history of hundreds of blockchains and found surprising similarities to non-blockchain open source software projects. While we certainly can't explain blockchain success in general, we believe that the same traits that lead to ideas that traits that partially explain their success. Based on this new perspective, we designed a new blockchain, Liquidum. It combines Chia's decentralized and sustainable consensus engine with Ethereum's flexible and widely adopted application layer.

2 Related Work

The question of why some blockchain ecosystems thrive while others fail can be viewed as a special case of the more general question why some open source projects gain adoption while others fail. Much has been written about this subject and upon review of the published material important themes emerged.

2.1 Leadership

Schweik and English [14] examined 174,000 open-source projects and found that 17 percent were successful. Almost half, 46 percent, of projects were abandoned before their first release and 37 percent of projects were abandoned after their first release. The projects that managed to be successful had characteristics in common, three of which were deemed "causal": (1) leadership by doing (the founders working more hours), (2) Clear vision, (3) well-articulated goals. Complimenting results were found by Mu, et [9] in that the most successful blockchains had leaders that have relation-oriented leadership behaviors that are open oriented and task-oriented leadership behaviors in the form of technical contributions.

2.2 Innovative Participants

Hippel [16] describes successful open source ecosystems as "innovation networks". Hippel contends that these networks can emerge when three conditions are met: (1) at least some users have sufficient incentive to innovate, (2) at least some users have an incentive to voluntarily reveal information sufficient to enable others to reproduce their innovations, and (3) user self production can compete with commercial production and distribution. Berkhout M., et al [2] apply the Open-source Software Ecosystem Health Operationalization (OSEHA) framework to cryptocurrency ecosystems to get insights into the ecosystem's value, longevity and propensity for growth. It found that the "healthiest" cryptocurrency ecosystems were best predicted by the number of new projects started for a cryptocurrency and the productivity of developers within a cryptocurrency ecosystem measured by lines of code added over time. Using this approach the authors found correlation with the success of Bitcoin, Ethereum and Ripple and the metrics mentioned earlier.

2.3 Open Technical Strategy

A similar observation is made by Kim M., et al [7] when they list "modifying the operation base" as one of their steps to achieving sustainable cryptocurrency ecosystem growth.

2.4 EVM Based Blockchains

The value of the EVM has been recognized by others, many blockchains have selected it as the based for their application layer. Avalanche, launched in 2018, is a blockchain that combines the EVM with a novel quorum based consensus protocol [13]. Since it's launch it has gained success as a target of dApp developers as well as a store of value [4]. Qtum, launched in 2017, combined the UTXO model from Bitcoin, a Proof-of-Stake consensus protocol and the EVM to create a novel blockchain. QTUM has achieved success as a value store [5].

2.5 Chia

Chia uses a Proof-of-Space mechanism in its consensus protocol [11], a new virtual machine called the "chia lisp virtual machine" for its application layer and a UTXO model of state similar to Bitcoin. Liquidum builds on this by using an account model for state and using the EVM for its application layer. Other projects have started utilizing Chia's technology, however, so far these have centered around rebranding or changes to initial pre-mine configurations.

2.6 Spirit of Open Source

In his seminal essay "The Cathedral and the Bazaar" [12], Eric Raymond explains the success of the Linux kernel by highlighting its open "Bazaar" development model as opposed to a closed/"Cathedral" development model. One of the elements stressed in his work

3 Ecosystem Potential

During our study of blockchains we developed a construct called "Ecosystem Potential" that we contend better captures the qualities that explain success of than current frameworks. Ecosystem potential is composed of three facets: Originality, Entrepreneurship, Einstellung. Each facet can be broken down further into components.

3.1 Originality

Originality measures the value that is being added to the network by updates to the blockchain model. Originality has two components: (1) novelty - the innovativeness of the change, (2) impact - a weight that captures how important the problem the changes are solving. Nearly every blockchain that has gained significant adoption had high originality when it launched. Bitcoin was the first blockchain powered cryptocurrency; Ethereum introduced application layers; Chia introduced a Proof-of-Space mechanism to power consensus.

3.2 Entrepreneurship

Entrepreneurship measures the ability of network participants to innovate and create value on or within the blockchain ecosystem. It is composed of two components: (1) full

node mechanics - the ability for full nodes operators create value, (2) application layer - the ability for developers to develop and deploy dApp on the network.

3.3 Einstellung

Einstellung is a german word that refers to the predisposition to solve a given problem in a specific manner even though better or more appropriate methods of solving the problem exist. We use this term (in the opposite sense) to measure the ability of a blockchain ecosystem to adapt and keep pace with its communities needs as well as the evolution of blockchain technology. It has two components: (1) governance strategy - how well does the project collect and incorporate feedback from its constituents, (2) development strategy - the ability of the ecosystem to come up with and agree on practical solutions to technical problems.

3.4 Blockchain Ecosystem Potentials

We rated some of the most popular blockchains based on these facets. Each blockchain was assigned a score between 1 and 10 for each facet for a total possible score of 30 for ecosystem potential. Below you will find the results:

Blockchain	**Originality**	**Entrepreneurship**	**Einstellung**	**Ecosystem Potential**
Ethereum	9	9	7	25
Chia	7	4	2	13
Binance Smart Chain	7	9	6	22
Internet Computer	8	8	7	23
Solana	8	7	7	22
Bitcoin	10	4	5	19
Filecoin	10	5	3	18
Neo	6	7	3	16
Cardano	8	3	4	15
EOS	7	5	3	15
Algorand	9	4	1	14
Hashgraph	8	2	2	12
Ethereum Classic	2	5	2	9
Ripple	3	1	2	5

Table 1: Our Ecosystem Potential Analysis Results

4 The Liquidum Network

Liquidum combines of the consensus layer from Chia and the application layer from Ethereum. In this section, we describe how it is built from these consituent parts, focusing on the important details and differences. We assume the reader is familiar with Ethereum and the EVM, and somewhat familiar with Chia's consensus and mining. After we have desribed what Liquidum is, we use the following section to explain how its design was guided by ecosystem potential.

4.1 The Blockchain

Like all blockchains, Liquidum nodes grow a shared data structure consisting of a chain of blocks. These blocks contain all data necessary to operate the network and validate future blocks, including transactions and state. We won't explain blockchains in detail, as there are already excellent resources on the subject [17] [1].

4.1.1 Chain Structure

Like Chia, only a subset of Liquidum blocks contain transactions. These *transaction blocks* are distinguished from non-transaction blocks in how they are numbered and processed. Only transaction blocks affect the blockchain state, while non-transaction blocks are only used for consensus. Non-transaction blocks are necessary because of the way that creation and finalization in Chia's consensus overlap between consecutive blocks (see [11] for details).

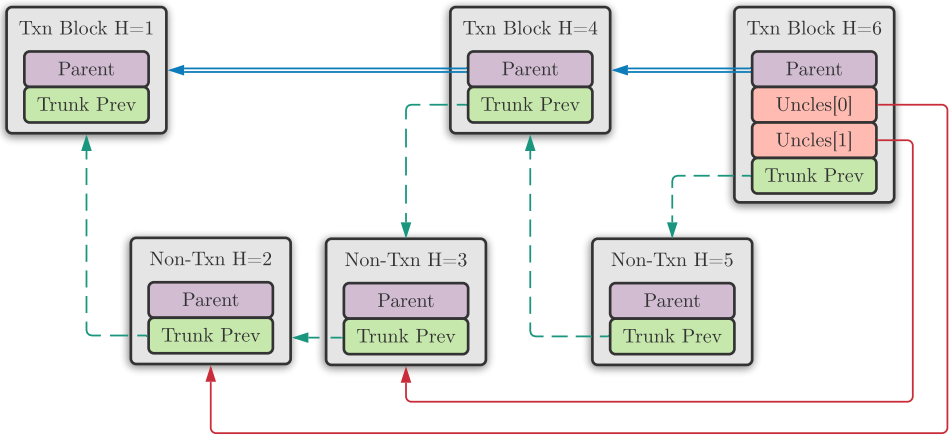

Figure 1: Transaction and Non-Transaction Blocks

Figure 1 shows how blocks are connected via various hashes to form a chain. The dotted green line from *trunk prev* pointers to a previous block forms the *trunk* chain. The trunk chain is equivalent to Chia's reward chain and contains the entire path from each valid block to the genesis block. The trunk chain has a subchain consisting of all transaction blocks, connected in the figure by the double blue lines. Each block's *parent* hash points from it to the previous transaction block. Only this chain is visible to transactions via EVM instructions, and only these blocks affect application state.

Lastly, transaction blocks also contain a list of zero or more *uncle* block pointers, shown as solid red lines. These play the same role as reward claims in Chia, rewarding the farmers of previous non-transaction blocks. The uncles list must contain the set of non-transaction blocks between the prev-prev transaction block and the prev transaction block, as shown in the figure. Unlike Ethereum, this list must contain exactly this set of blocks and is not a function of block propagation time.

4.1.2 Block Structure

Liquidum blocks consist of a header and a body, both of which are required for validation. The block header combines an Ethereum header with a Chia reward chain block and a few additional fields related to consensus. The block body is identical in structure

and semantics to Ethereum's block body. Table 2 lists the fields in the Liquidum block header.

Field	Description
Parent Hash	Hash of previous transaction block
Uncle Hash	Hash of *Uncles* list, see Fig 1.
Coinbase	LQM address receiving reward and fees for this block.
State Root	Same as ETH
Transactions Root	Same as ETH
Receipts Root	Same as ETH
Logs Bloom	Same as ETH
Difficulty	Same meaning as ETH, computed according to Chia rules
Number	Same as ETH, defined as parent block number + 1. Visible to EVM.
Gas Limit	Same as ETH with EIP-1559
Gas Used	Same as ETH
Timestamp	Same as ETH
Extra Data	Same as ETH
Mix Hash	Reserved, must be zero. Contained proof of work hash in ETH.
Nonce	Reserved, must be zero. Contained proof of work in ETH.
Base Fee	Minimum gas price, burned on transaction inclusion. EIP-1559
Governance Data	Used for governance related votes and messages. See section 4.6
Proof of Space and Time	Modified Hellman proof of space and VDF challenge proofs. In Chia, this is the *Unfinished Reward Block*.
Signature	Farmer signature of block data, everything above in this table. Similar to the foliage signatures in Chia.
Trunk Prev Hash	Similar to the previous block hash in Chia.
Trunk Validation Data	Similar to the reward chain infusion proofs in Chia.
Block Validation Data	Similar to the challenge chain infusion proofs in Chia.

Table 2: Liquidum Header Data. Purple rows are from Ethereum, green from Chia.

Ethereum header fields, purple lines in the table, differ only a little from their Liquidum counterparts. Mix Hash and Nonce are unused but reserved for compatibility and future use. Also, Liquidum uses the EIP-1559 fee structure which became active on Ethereum mainnet after the *London* hard fork in August, 2021.

We have added a new field for governance related metadata, denoted by a blue-green in the table. This field will be used for voting and messaging. See section 4.6 for details.

The green colored fields are taken from Chia, which slightly more modification than the Ethereum fields. Details are in the table, but most fields have a one to one mapping with fields in the Chia blockchain. The main difference between Liquidum and Chia block headers is that there are no Foliage blocks in Liquidum. The Ethereum header data fulfills the same purposes as the Foliage in Chia. Instead, we include the farmer signature directly in the header, and eliminate the notion of having two separate chains to distinguish transactions from consensus.

Lastly, we should mention that the Liquidum block contains the same transaction data that would be included in an Ethereum block, outside the header. The arrangement of the complete block is shown in Figure 2.

4.2 Transactions and State

The Liquidum transaction layer is nearly identical to Ethereum's. We will give a light overview of transaction processing but assume the reader is familiar with Ethereum. Liquidum has a *state* consisting of all balances, contract storage, contract code, account nonces, and related information. The state is transformed by transactions according to well-defined rules. We can think of each transaction as a function that takes as input the previous state value and produces as output a new, transformed state value.

In Liquidum, each block contains an ordered list of transactions which are commited-to via the *Transactions Root* header field. The binary structure and meaning of each

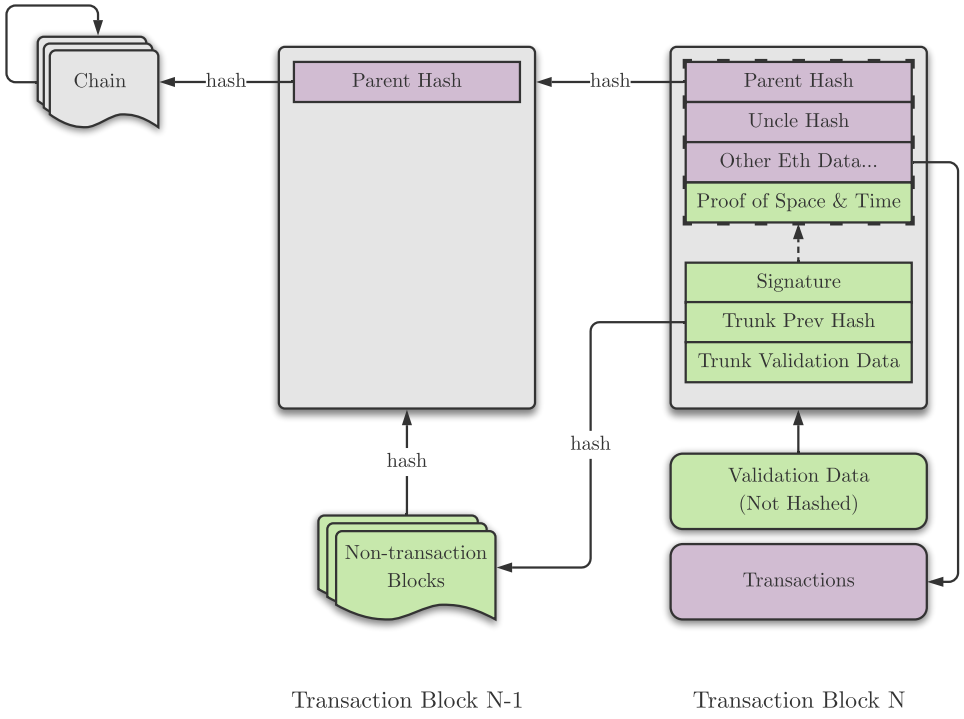

Figure 2: Liquidum Full Blocks

transaction are identical to Ethereum transactions. The state at block 0, just before the genesis block, consists of all balances and contract storage locations having a value of *0*. Each transaction in the genesis block transforms the state consuming the output of the previous transaction in the list, and in turn having its output consumed by the next transaction. The final state at the end of the block consists of the repeated application of each transaction, in order: $S_{final} = T_N(T_{N-1}(\dots T_2(T_1(S_{initial}))))$.

Likewise, subsequent blocks after gensis transform the state in the same way. We can say that each block N defines a *state of the blockchain*, which is just the result of transforming the initial genesis state with every transaction from block 0 to block N. For example, an account's balance is just the sum of all the balance changes for that account in every transaction from genesis to the current block.

4.3 Nodes

The Liquidum network is a distributed system of nodes that communicate with one another and maintain internal state. Each node has a type which determines what role it plays in the network. Honest nodes communicate with one another via a precisely specified protocol, and change their internal state according to well defined rules.

Not all nodes are the same. As we will describe in the next section, Each node maintains its own view of the blockchain and the Liquidum state.

4.3.1 Full Node

The full node maintains the state of the blockchain, collecting transactions and assembling blocks for future inclusion in the chain.

In our implementation, the full node is a modified version of the go-ethereum (geth) software that uses chia-blockchain as a library for header validation. We have heavily modified the block validation and mining portions of geth to mine and validate blocks according to the "hellman" consensus protocol.

4.3.2 Timelord

Timelords maintain VDF chain state and compute VDF functions according to the consensus rules. The Liquidum timelord protocol and implementation is an essentially unmodified version of the Chia timelord. Only one honest Timelord is necessary for network availability, and the Timelord cannot unilaterally violate any safety or consistency properties of the network. Timelords are not compensated in any way by the protocol. In addition, there is some advantage to operating a Timelord on expensive, specialized hardware, perhaps even an ASIC. Hence the network will likely only have a small number of Timelords operated by either large miners or the Liquidum founders.

For more information, see the chia new consensus paper [11].

4.3.3 Farmer and Harvester

The Farmer and Harvester node types remain mostly unmodified from Chia. Liquidum farmers listen for VDF challenges, and send them to their connected harvesters. Harvesters listen for challenges and look up proofs in their plots whenever the plot passes a challenge specific plot filter. In response to Chia community feedback, the Liquidum harvester has been modified to reduce the worst case latency for proof lookups and adds increased monitoring and logging on reward-critical paths. This should reduce the risk of farmers missing out on rewards by responding too quickly to network challenges. We have used these modifications in our Chia farming operation to farm over ten thousand plots with a single harvester and a p99 latency under $2.5s$. We plan to upstream our changes to the *chiapos* repo.

4.4 Consensus and Mining

Liquidum uses a modifed version of the proof of space and time consensus algorithm from Chia, as described in [11]. The initial software implementation uses a forked version of the the Chia full node reference code for mining and block header validation [6]. Here we provide a short overview of Mining and consensus but assume the reader is familiar with Chia.

4.4.1 Farming Overview

Liquidum miners are called *farmers*. A Farmer join the network by generating large unique *plot files* with a special structure based on the Hellman proof of space construction TODO cite. The Farmer connects to the network and listens for *challenges* sent by timelords. When the farmer recieves a challenge, it looks up *proofs*, which are solutions to this challenge, in each of its plot files. For each proof that the farmer finds, it computes a *quality*. If a proof has a "good" enough quality the farmer can use it to create a block and receive a reward. The "goodness" of a quality is defined arbitrary to control the number of eligble proofs network-wide, and hence the block rate. The farmer fills a new block with transactions in its mempool, includes the proof, signs the block, and broadcasts it over the network. The special structure of the plot file makes it possible for anybody to verify the correctness and quality of a proof without access to the plot file.

Because each proof depends on a specific challenge, the farmer can only generate the proof **after** receiving the challenge. Although we don't describe it here, farmers must also *infuse* their blocks to prove that they generated the proof **before** some later point in time. Together the challenge, infusion, and the proof of space show that the farmer stored a plot file at a particular point in time. Hence the name "proof of space and time".

We use the same plot file structure, plot filtering, and proof eligibility rules as Chia, with the addition of a Liquidum specific plot type. We believe that retaining compatibility with Chia is important because thousands of Chia farmers already know and love its farming mechanic. We also want to maintain compatability with tools built for Chia farmers, so that it will be extremely cheap for developers to make these tools also work with Liquidum.

4.4.2 Block Rewards

Farmers are paid a reward for each block that they farm. The rewards incentivize them to process transactions and secure the network. Initially, block rewards in Liquidum will be constant, with no scheduled decrease. We take no strong position on whether the native fee token should be inflationary or deflationary, and leave that question open to future community led decision.

Liquidum does not distinguish between farmer rewards and pool rewards. In Chia, $\frac{7}{8}$ of each block reward is sent to a pool address, with the remaining $\frac{1}{8}$ sent to an address chosen by the farmer. This design is meant to mitigate *block withholding attacks* in which pools intentionally direct their mining resources (netspace) toward a victim pool. The attacker earns partial rewards from the victim pool but withholds full blocks to reduce the victim's revenue, forcing the victim to fold.

While we agree that this attack is potentially damaging to decentralization, we do not believe that the protocol rewards are the right tool to mitigiate it. A recent block withholding attack caused a small but growing Chia pool called Maxiopool to shut down, showing that Chia's hard-coded payout does not adequately protect small pools [8]. Pools should choose for themselves how to structure their payouts to prevent these attacks. Over time the community will learn how to effectively mitigate attacks like the one against Maxiopool. Hence, Liquidum pays 100% of the block reward to a single address.

Unlike Ethereum, uncle blocks receive the same reward as ordinary blocks. This is because uncle blocks play the role of Chia non-transaction blocks, and are just as necessary for consensus as non-uncles.

4.5 Chia Plot Compatibility

Liquidum supports existing Chia plots, meaning that Chia farmers can immediately begin farming and earning rewards on Liquidum without regenerating plot files ("replotting") or increasing their operating costs. This will bootstrap the network by leveraging the growing ecosystem of Chia farmers and services.

Allowing Chia plots opens the network up to possible 51% attacks because of the relatively large Chia netspace. This is a common problem for consensus forks. For example Ethereum Classic has suffered several 51% attacks as a result of its reusing Ethereum's proof of work algorithm [15]. We leave the option open to migrate away from Chia plots using either future protocol changes or economic incentives, for example giving larger block rewards and weight to native plots.

Our approach to preserving compatibility with Chia plots is interesting and worth describing in technical detail. For background, Chia has two plot types that differ in their reward semantics: *Public Key* aka *OG* plots, and *NFT* aka *pool* plots.

4.5.1 Public Key Plots

Public key plots have a BLS public key embedded within them in such a way that it is possible to check the value of this public key given any proof from the plot. In Chia, this public key receives the $\frac{7}{8}$ pool reward for any block reward earned using the plot. Recall that Liquidum uses ordinary Ethereum addresses, which do not have BLS public keys (Ethereum 1.x uses secp256k1). It would not be possible for farmers to prove that a Liquidum payout address is controlled by the pool public key. Fortunately, our change to block rewards solves this problem automatically. We ignore the plot public key and give 100% of the block reward goes to the farmer's chosen address. This may seem problematic, but makes no practical difference. The vast majority of public key plots were generated with a public key controlled by the farmer itself, and sending all rewards to the farmer is equivalent for these plots.

4.5.2 NFT Plots

NFT plots are more complicated. These plots are derived from a specific Chia payout address, also called a "puzzle hash". On Chia, these are typically the address of an on-chain smart contract (NFT) designed to make secure pooling possible. The NFT *owner* designates a flexible payout address to which block rewards for the plot are forwarded. The payout address can only be changed after a public announcement and lengthy timeout period (20-30 minutes). The flexible address allows the farmer to switch pools or mine for itself without replotting. The timeout period gives pools advanced warning that pool participants will exit the pool so that they can stop paying partial rewards.

Recall that Liquidum uses the EVM, has a completely different application layer than Chia. The smart contract addresses embedded in existing Chia plots are meaningless to the EVM. We solve this by modifying the EVM to include a special contract creation mechanism specifically designed to build bridges between Chia and Ethereum. In order to use existing Chia NFT plots, farmers must issue a one-time Liquidum transaction to register their NFT. They make one transaction per NFT, which for most farmers will be shared by all of their plots.

The details of this registration process are shown in Figure 3. The Farmer submits a Liqiudum transaction with the NFT's original compiled CLVM code, a signed message indicating that the plot owner wishes to register the Chia NFT on liquidum, and new EVM code to deploy. Liquidum's EVM has a special precompiled contract `CREATE_CHIA_NFT` that takes these inputs, verifies the signature and CLVM code hash, and creates a new EVM contract whose code is the submitted EVM code and whose address is derived from the CLVM code puzzle hash. The block rewards which contain the puzzle hash are sent to the same derived address.

The security of this process depends on a few tricks. The CLVM NFT code contains the BLS public key of the NFT owner, which is used to authorize changes to the payout address. We use this public key to verify that the registration transaction was submitted by the actual NFT plot owner, since nobody can forge a signature with this public key. The address derivation is arbitrary, so we can choose it to be both fast (for block validation) and to prevent possible collisions with Liquidum contracts created by `CREATE` and `CREATE2`. It is possible to craft CLVM code with multiple BLS public keys, so that multiple calls to `CREATE_CHIA_NFT` would create colliding Liquidum contract addresses. We currently prevent this by disallowing duplicate registrations with the same resulting address. Tracking these duplicates requires the precompiled creation contract to store a new state entry for each registered NFT, increasing registration and block validation. We are looking for ideas from the community to solve this more efficiently.

The mechanism described here is actually quite powerful and general, with applications beyond plot registration. It is a tool tool to give Liquidum users cryptographic

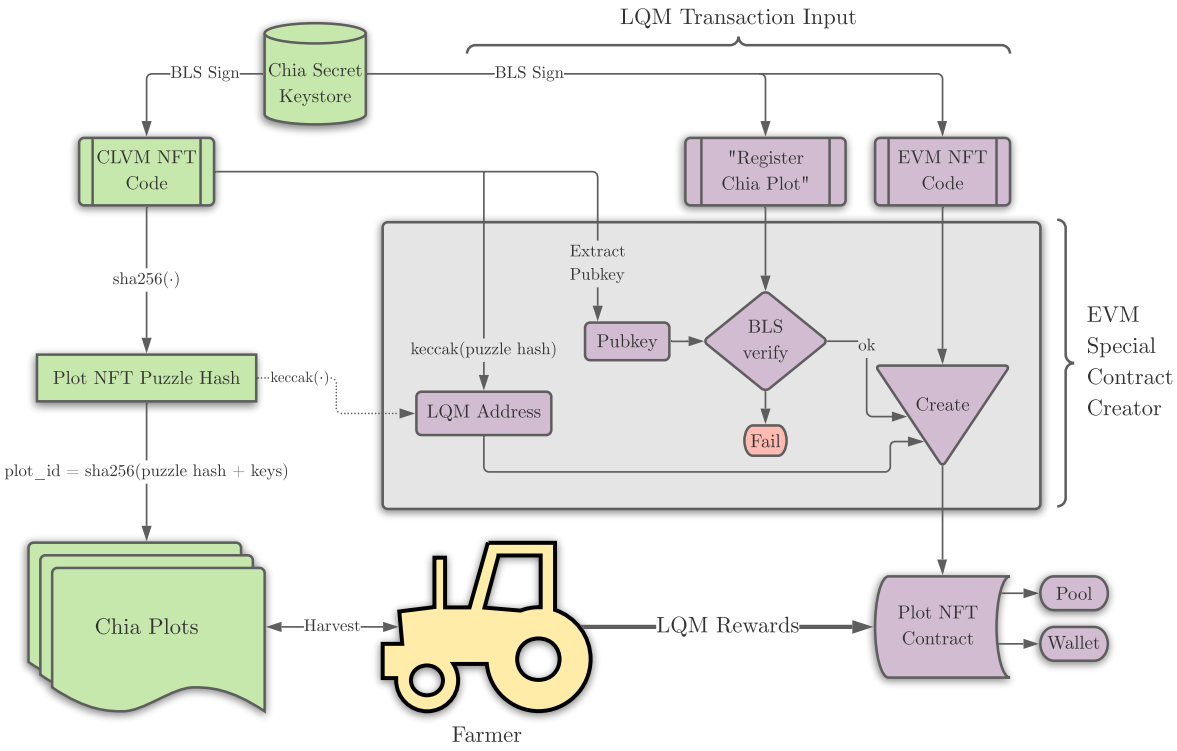

Figure 3: Chia Plot NFT Compatibility

control over resources they control on the Chia blockchain, or any other chain that uses BLS signatures. We are exploring possibilities to extend the use cases, and expect that dApp developers will build things we couldn't imagine ourselves.

4.6 Governance

TODO
[3]

4.7 Security

Liquidum's security relies on Chia's security. Because the Liquidum blockchain includes a Chia blockchain, any attack on Liquidum would also be possible on Chia. Specifically, there is a mapping from Liqudum blockchains to Chia blockchains such that for any valid Liquidum blockchain, there is a unique corresponding valid Chia blockchain that can be constructed from the Liquidum blockchain. This mapping relies only on standard assumptions about cryptographic primitives, that hashes are collision and preimage resistant, and signatures are non-forgeable. Further, the peak weight of the two chains matches. Hence, any attack on the Liquidum chain implies an attack on the Chia chain. Future versions of this paper will include a complete proof.

4.8 Pooling

Liquidum provides mechanisms for creating secure mining pools based on ideas from Chia and Ethereum.

5 Liquidum's Ecosystem Potential

5.1 Originality

There are few domains in the blockchain space where novel innovations have the potential for high impact. Two of which are: (1) the mechanism used by the consensus protocol to achieve Sybil attack [19] resistance and (2) the design of the application layer. By combining Chia's Proof-of-Space mechanic with Ethereum's EVM based application layer Liquidum strives to achieve value larger than just the sum of it's parts. Borrowing the concept "Emergence" [18] from systems theory, we contend that the combination of a sustainable Proof-of-Work mechanic with an established, developer friendly application layer will achieve an Originality score better than just the novel combination of the two. To illustrate this point take our approach to pooling: the adoption of Chia's Proof-of-Space mechanic allows miners to join pools on chain without having to download any custom software while the adoption of the EVM allows pool operators to implement their own pool participation rules in Solidity. This is just one of many "synergies" that we expect from this combination. For these reasons we have assigned a rating of 10 for Originality to Liquidum.

5.2 Entrepreneurship

Liquidum maximizes the potential for entrepreneurship with a two pronged approach: (1) invest in a mining mechanic that invites innovation by network participants and (2) develop an application layer that supports the widest possible set of applications.

5.2.1 Mining Mechanics

Many blockchains have settled on Proof-of-Stake as a replacement for traditional Proof-of-Work to mitigate the network's excessive electricity utilization, however, we believe this unnecessarily cuts out an important avenue for network participant entrepreneurship. Proof-of-Stake caps the innovation possible for miners. The "unbounded innovation potential" nature of Proof-of-Work is an important motivator for innovative network participants. Proof-of-Space captures that same unbounded potential of traditional Proof-of-Work without the undesirable electricity untilization properties. There is already evidence that innovation is taking place (see madMAx43v3r chia-plotter [20]).

5.2.2 Application Layer

Ethereum and its developer toolchain ecosystem marked a breakthrough in developer productivity in blockchain technology. Complex decentralized applications, dApps, like UniSwap, Compound and OpenSea have been deployed with great success due in large part to the accessibility of Solidity, Web3, the EVM and other Ethereum technologies. By adopting these technologies Liquidum is poised to inherit that same entrepreneurial vigor. However, many would be product entrepreneurs still find blockchain technology inaccessible due to the "incomplete solutions" offered by blockchains. Solutions to problems like persistent off-chain state and off-chain data oracles presently can be overcome but usually with great effort and without certainty that such solution are the "correct" ones. This severely limits what product innovators perceive blockchains can do. We will

address these shortcomings by cataloging common platform limitations and developing official solutions that will fill in the gaps and enable product innovators to develop more and more sophisticated products on Liquidum.

5.3 Einstellung

Liquidum's approach to governance and development process is largely modeled after the most successful open source organizations to date. The Linux Foundation, the Ethereum Foundation and Red Hat, Inc (to name just a few) all have differing purposes and "products" but they have one important thing in common: they created enormously popular open source software projects despite not having a "specific vision" of what should exist. Instead of a traditional "top down" method of organizing typically seen in large organizations these entities function as aggregators and implementers of good ideas. Those ideas are sourced not from "management" but from organization participants that recognize problems and suggest solutions. Solutions are debated and iterated on until there is relative agreement on the path forward. Let's explore the specifics.

5.3.1 Development Process

For our community and development process, we embrace lessons about project growth from traditional software engineering. Simplicity is more important consistency, completeness, or correctness ("Worse is Better"). Software should be developed in the open with an inclusive, bottom-up process ("The Bazaar model"). No single innovation or strategy will by itself lead to a tenfold increase in developer productivity, but blockchain software is distinct from more traditional fields in having more removable complexity ("No Silver Bullet"). We also describe future directions to increase adoption via scaling, tooling, and economic incentives.

5.3.2 Governance

All discussion will take place in the open: on forums, in discord or on Github. "Product direction" is sourced from the community: network participants can suggest ideas wherever they want, so as far as to facilitate discussion. Foundational changes to Liquidum such as changes to core protocols, client APIs and contract standards will be handled through Liquidum Improvement Proposals, LIPs. Voting will be handled on-chain using a special header field that allows miners to cast their votes in full public view. However, these votes are just to maximize the transparence of what the community thinks about a particular issue. The final decision of what will actually get implemented is still up to the core developers.

6 Future Work

* Liquidum achieves decentralization and high ecosystem potential, but we have future work planned to improve scalability. We want to buid governance around ecosystem growth, and we want to improve the end-user and developer experiences

6.1 Scalability

6.1.1 Managed State Services

Full nodes and farmers can share state.

6.1.2 Sharding and ETH 2.0

We could maybe use proof of space and time instead of proof of stake.

6.1.3 Commutative transactions and DAG fork-choice rules.

6.2 Governance

6.2.1 Soft Work Weighted Voted

6.3 User Experience

6.3.1 TODO

6.4 Developer Experience

6.4.1 Asynchronous Programming

7 Conclusion

Liquidum was founded to achieve a vision: To create a platform for distributed applications that is sustainable, censorship resistant, and focused on developer productivity. Optimized for these qualities, Liquidum is poised to be a leader in the blockchain space.

References

[1] Andreas M. Antonopoulos. Chapter 7. the blockchain. In *Mastering Bitcoin: Unlocking Digital Crypto-Currencies*, chapter 7. O'Reilly Media, Inc., 1st edition, 2014. https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch07.html.

[2] M. Berkhout, F. van den Brink, M. van Zwienen, P. van Vulpen, and S. Jansen. Software ecosystem health of cryptocurrencies. In K. Wnuk and S. Brinkkemper, editors, *Software Business*, volume 336. Lecture Notes in Business Information Processing, Springer, Cham, 2018. https://doi.org/10.1007/978-3-030-04840-2_3.

[3] Vitalik Buterin. Notes on blockchain governance. https://vitalik.ca/general/2017/12/17/voting.html, 2017.

[4] Coinmarketcap. Avalanche market cap. https://coinmarketcap.com/currencies/avalanche/, 2021. [Online; accessed 21-August-2021].

[5] Coinmarketcap. Qtum market cap. https://coinmarketcap.com/currencies/qtum/, 2021. [Online; accessed 21-August-2021].

[6] Liquidum Contributors. Liquidum fork of chia full node. https://github.com/liquidum-network/chia-blockchain, 2021.

[7] Moon Soo Kim and Jee Yong Chung. Sustainable growth and token economy design: The case of steemit. *Sustainability*, 11(1), 2019. https://www.mdpi.com/2071-1050/11/1/167.

[8] Maxiopool. We're shutting down. https://maxiopool.io/shutting-down, 2021.

[9] Wen Mu, Yiyang Bian, and J. Zhao. The role of online leadership in open collaborative innovation: Evidence from blockchain open source projects. *Industrial Management & Data Systems*, ahead-of-print, 10 2019. https://doi.org/10.1108/IMDS-03-2019-0136.

[10] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf, 2009.

[11] Chia Network. Chia consensus. https://www.chia.net/assets/Chia-New-Consensus-0.9.pdf, 2021.

[12] Eric Raymond. The cathedral and the bazaar. *Knowledge, Technology & Policy*, 12(3):23–49, Sep 1999.

[13] Team Rocket, Maofan Yin, Kevin Sekniqi, Robbert van Renesse, and Emin Gün Sirer. Scalable and probabilistic leaderless bft consensus through metastability. https://arxiv.org/pdf/1906.08936.pdf, 2020.

[14] Charles M. Schweik and Robert C. English. *Internet Success: A Study of Open-Source Software Commons*. The MIT Press, 2012. http://www.jstor.org/stable/j.ctt5vjpw8.

[15] Zack Voell. Ethereum classic hit by third 51% attack in a month. https://www.coindesk.com/ethereum-classic-blockchain-subject-to-yet-another-51-attack.

[16] Eric A. von Hippel. Horizontal innovation networks - by and for users, May 2009. https://ssrn.com/abstract=1411719.

[17] Wikipedia contributors. Blockchain — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Blockchain&oldid=1033575964, 2021. [Online; accessed 5-August-2021].

[18] Wikipedia contributors. Emergence — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Emergence&oldid=1038341456, 2021. [Online; accessed 12-August-2021].

[19] Wikipedia contributors. Sybil attack — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Sybil_attack&oldid=1036212871, 2021. [Online; accessed 12-August-2021].

[20] Max Wittal. chia-plotter (pipelined multi-threaded). https://github.com/madMAx43v3r/chia-plotter, 2021.

A Appendix: Blockchain Analyses

Ethereum and its developer toolchain ecosystem marked a breakthrough in developer productivity in blockchain technology. Complex decentralized applications, dApps, like UniSwap, Compound and OpenSea have been deployed with great success due in large part to the accessibility of Solidity, Web3, the EVM and other Ethereum technologies. However, the Proof-of-Work, PoW, mechanism employed by Ethereum represents a major hurdle to scaling the network. The Ethereum network currently utilizes as much electricity as the country of Romania (55.01 TWh) [1]. There is a plan to mitigate this by migrating Ethereum to Proof-of-Stake (PoStake), but this introduces new problems. One of the most problematic issues is that there is no way for a miner to participate without "Registering". In order for a miner to participate they must first have ether which in most cases requires KYC. This introduces many avenues for gatekeeping or censorship. PoSpace does not suffer from these types of problems. By combining Chia's PoSpace with Ethereum's developer ecosystem Liquidum captures the best of both worlds.

A.1 Protocol Comparisons

A.2 Liquidum vs Bitcoin and Ethereum (Istanbul)

- + PoSpace is energy efficient
- + PoSpace uses different resources. Anyone can participate, not just ASIC miners.
- - More complex, more difficult to implement

A.3 Liquidum vs Ethereum (Gasper)

- + PoSpace is permissionless vs PoStake requires participants to use onramps that could be controlled by governments or biased interests.
- + Use of VDFs less susceptible to "long range attacks" vs Gasper
- + More straightforward to scale
- + No liveness requirement, assumes less about network participants being online
- + No need for checkpoints with 2/3 consensus
- - No finality, block finalization is probabilistic
- - Longer transaction wait times

A.4 Application Layer

The other critical aspect of a blockchain is it's "application layer". This refers to the platform offered by a blockchain to developers to create, deploy and host distributed applications, "dApps". Every blockchain's application layer has different characteristics based on the priorities of it's designers. For example Bitcoin, with all its breakthrough contributions to crypto, has a very limited application layer. It's scripting language is not Turing complete and is only really useful for performing light control flow during transaction execution. Liquidum's application layer is based on the Ethereum Virtual Machine (EVM) and it's corresponding toolchain. Let's compare Liquidum's application layer to Chia's.

A.5 Chia

Programs for Chia are executed on the Chia Lisp Virtual Machine (CLVM). The programs are written in a high level language called "ChiaLisp" and compiled to "CLVM Bytecode" which the CLVM executes. The computational model for CLVM follows other lisp dialects and represents programs as a binary tree. The root of the tree is the least nested object in the program tree, with inner function calls embedded recursively inside of it. Program trees are evaluated by first evaluating the leaf nodes, then their parents, recursively. Like the EVM, the CLVM maintains transient memory which does not persist between transactions. However, unlike the EVM, the CLVM follows the UTXO model. This means that ChiaLisp handles persistent state in a much less direct way. Each program on the CLVM can be thought of as an "Unspent Coin" that requires a special ChiaLisp program called a "Solution" which solves the coins "Puzzle", another ChiaLisp program which specifies the behavior of the coin when it is spent. When a coin is spent it is destroyed but it can generate new coins. When a developer wants to create a dApp that maintains state on chain they will have to program it as a self-propagating coin that regenerates itself with a new version of it's state every time it is spent. Additionally, since the CLVM/ChiaLisp is new, the ecosystem around it is still immature. Common design patterns, open source libraries, IDEs and other developer tools are still lacking.

A.6 Examples

Let's take look at how different application layers look in practice by implementing a very simple program that stores an integer and has the ability to calculate the sum of that number and any number passed to it by a different caller. For a baseline we will use a javascript class that has this functionality.

```
class FixedSummer {
  constructor(initVal) {
    this.storedVal = initVal;
  }
  getSum(otherVal) {
    return this.storedVal + otherVal;
  }
}
```

A.6.1 Liquidum

```
contract FixedSummer {
    uint storedVal;
    constructor(uint initVal) {
        storedVal = initVal;
    }

    function getSum(uint otherVal) public view returns (uint) {
        return storedVal + otherVal;
    }
}
```

A.6.2 Chia

In ChiaLisp state is handled by "currying". The idea is to pass in arguments to a program before it is hashed. When you curry, you commit to values so that the individual running the program cannot change them.

```
;; utility function used by curry
(defun fix_curry_args (items core)
  (if items
    (qq (c (q . (unquote (f items)))
      (unquote (fix_curry_args (r items) core))))
     core))

;; (curry sum (list 50 60)) => returns a function
;; that is like (sum 50 60 ...)
(defun curry (func list_of_args)
  (qq (a (q . (unquote func))
    (unquote (fix_curry_args list_of_args (q . 1))))))
```

Listing 1: ChiaLisp example

Every blockchain achieves byzantine fault tolerance[cite definition] by implementing a "consensus protocol". Each protocol is composed of rules and algorithms that govern things like leader election, block proposal and block finalization. Liquidum's consensus protocol takes inspiration from many important blockchains but introduces some improvements. Let's briefly examine a few important ones before comparing them against Liquidum.

A.7 Bitcoin

Bitcoin's consensus protocol is called "Nakamoto Consensus" and is composed of many elements but the two that define most of its behavior are the proof-of-work (PoW) mechanism and the "longest-chain-win" rule. Summarized simply, this translates to an honest node adopts the longest PoW chain and attempts to mine a new block that extends this chain. A block is considered finalized after it is buried sufficiently deep in the chain [citation].

A.8 Ethereum

Ethereum is currently in the process of radically changing it's consensus protocol so we will examine both versions. For convenience I refer to the original protocol using ethereum's fork name "Istanbul" and the new protocol by it's name "Gasper".

A.8.1 Istanbul

Istanbul's consensus is similar to bitcoin but instead of using the height of the block to calculate the "longest" chain Istanbul determines the longest chain based on the total difficulty, which is embedded in the block header. Ties are broken randomly. Istanbul also employs a PoW mechanism for proposing and verifying new blocks, however a different hash function is used (Ethash). Blocks are also considered finalized after they are buried sufficiently deep in the chain.

A.8.2 Gasper

Ethereum 2.0's consensus protocol called "Gasper" is different in nearly every aspect than Istanbul's. Instead of a PoW mechanism Gasper uses a Proof-of-Stake (PoStake) mechanism. Instead of miners solving computationally difficult puzzles and being rewarded the right to propose a new block (and earn rewards), they will "stake" ethereum and attest to a particular "view" of the blockchain. If they follow the protocol in good faith they will be rewarded with a "proposer reward" or "attester reward". If they behave in a malicious (byzintine) way they will have their stake "slashed". Additionally, instead of the "longest-chain-win" rule the "Latest Message Driven Greediest Heaviest

Observed SubTree" (LMD GHOST) is used. This rule chooses the chain with "the most activity" by measuring "activity" as weighted attestations [citation]. The selection of which validator gets to propose the block is done through a process called "Hybrid" LMD GHOST (HLMD GHOST) in which validators are arranged into "committees" and each committee proposes a block. Block finalization is done through a voting process similar to a traditional PBFT solution where 2/3 (by stake) have to agree on the new view of the blockchain to have the new block included in the chain.