Koinos Blockchain Framework Whitepaper

October 6, 2020

Blockchain: A New Computing Paradigm

Blockchains are a new method for performing computation that is still in its infancy but has massive disruptive potential as a result of its ability to establish digital ownership. This novel database structure, first pioneered in Bitcoin, enables developers to leverage decentralized networks of computers to perform computations in a trustless manner and incorporate digital ownership into their applications.

Operating Systems

As is usually the case in a new computing paradigm, in the early days there was no distinction between application and operating system (Bitcoin). Developers built their applications from the ground up, if only because no tooling existed. As more applications got built, it became possible to observe the common primitives that could be abstracted into a lower level program—an operating system—a simple, efficient, general purpose lower layer that would free application developers to focus on building a high level app without having to develop any low level code.

The creation of an operating system layer, however, is not the final stage in the evolution of a computing paradigm. While the early operating systems make it much easier for application developers to quickly build stable and secure applications that tap into a pre-existing user base, they eventually become feature-bloated, complex, and difficult to upgrade. Upgrading the system requires a system reboot and reloading process that becomes longer and more disruptive the bigger and more complicated it gets.

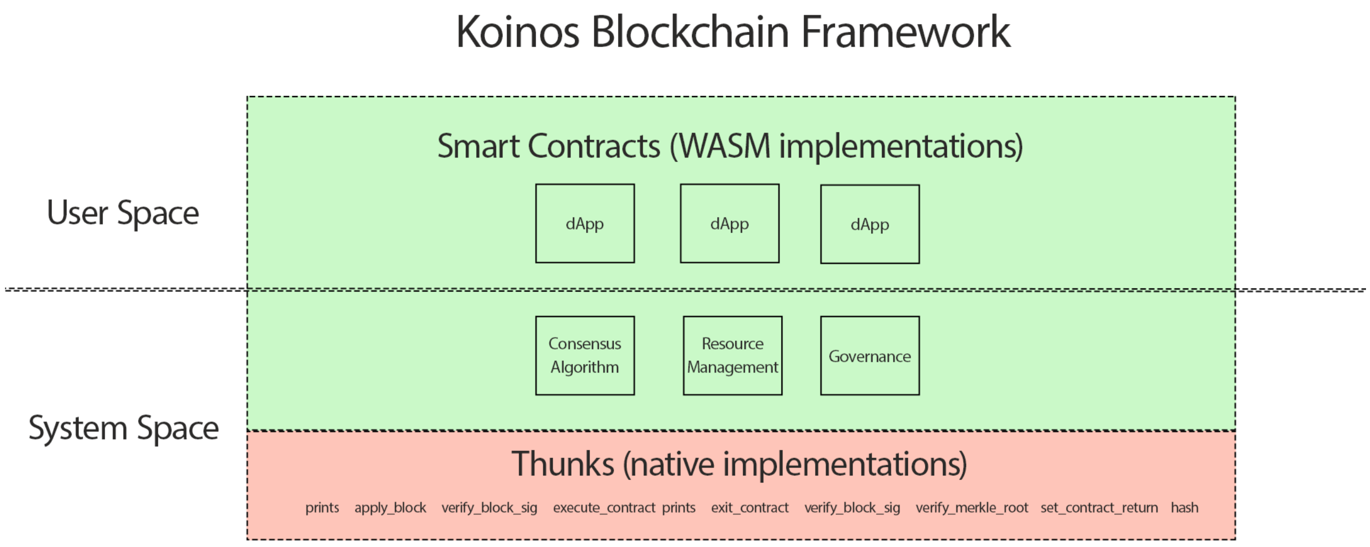
In the case of blockchains, these problems are amplified due to immutability; they are constantly growing in size. Further, because they are decentralized, upgrades must make their way through a political process which ends (under the best of circumstances) in the execution of a coordinated effort in which all of the computers stop running the old software and start running the new software (a new "fork") *at the exact same time*. This is far from easy, which is why calling it a "*hard*-fork" is especially apt.

Blockchain BIOS

The solution to this problem is an abstraction layer below even the operating system. An operating system for the operating system, designed not to make application development easier, but to make it easier to boot up and upgrade the operating system itself. While seemingly simple, this development is what enables the operating system to evolve because operating system developers can now rapidly respond to the changing needs of application developers by releasing software upgrades and bug patches to the network piecemeal while minimizing downtime.

Evolving Blockchain Framework

Koinos is the first blockchain with a BIOS equivalent which enables it to undergo rapid evolution. At the base of the Koinos stack is a blockchain framework that features a set of "thunks" that represent blockchain basics like contract input/output, getting parameters, writing to the database, etc. Above the framework is a system call layer that provides library-like support to smart contracts that can be upgraded in-band. System calls can be either thunks (native implementations) or smart contracts (WASM implementations).



Modular Upgradeability

This combination of thunks and system calls forms a high performance, vertically scalable, blockchain framework which can acquire any feature through smart contract modules running in the virtual machine. Because these smart contracts can be upgraded in-band, hardfork-like behavior change does not force a coordinated software upgrade on all users. This new capability that we call "modular upgradeability" resolves one of the biggest challenges facing existing blockchains: The difficulty and disruption of hardfork upgrades.

In traditional blockchains such as Bitcoin or Ethereum, a hardfork which implements new system features, or causes existing system features to act differently, necessitates all users to download new native code. In other words, the only way to implement new system feature behavior is to force a software upgrade on all users.

For a smart contract blockchain, there is no reason most upgrades need to be so disruptive. In Koinos, changes to system behavior can be distributed in blocks and transactions, identically to user smart contracts.

Governance

Any change to system rules involves a process of consent. In traditional blockchains, every user must consent to changes by installing a software upgrade. In Koinos, an on-chain governance mechanism will be needed to allow proposal and ratification of any changes to system-level behavior. The precise details of this mechanism are, as of this writing, not yet fully fleshed out. However here are a few principles:

- Block producers and long-term token holders will likely have governance roles.
- Changing key parameters like token minting rates may require a larger majority and/or wider group of stakeholders to opt-in.
- Checks and balances (i.e. block producers cannot propose certain kinds of changes, but can effectively veto any proposed change by refusing to include ratification transactions in blocks)
- Governance votes must be renewed (attempt to solve voter apathy problem)

Optimization

For performance reasons, many system features should be implemented natively (i.e. in C++). However, at any time, these native implementations can be replaced via the in-band upgrade mechanism. For this reason, when a contract accesses a system feature (by making a system call), a dynamic dispatcher checks a lookup table to determine the currently active implementation. This allows the performance of most system code being native, while preserving the flexibility of allowing any system feature to be upgraded at any time with WebAssembly code.

The Thunks

To understand the specific thunks Koinos was given, one needs to understand the basic purpose of the blockchain, which is to include and order transactions. That's why Koinos has apply_block and apply_transaction thunks. Since Koinos is a general purpose blockchain, it requires thunks for the input and output of smart contracts, such as logging (prints) and database management (db_put_object and db_get_object). Everything else has to do with authentication, optimization, etc.

Technically, those are the only thunks one needs in a blockchain framework because everything else can be handled by smart contracts running in the VM. But just because they could theoretically be handled by smart contracts doesn't mean this would be ideal, which is why Koinos has additional primitives that will be used constantly by the blockchain. For those primitives, and those alone, improved performance resulting from native C++ implementations will pay incredible dividends over the life of the protocol.

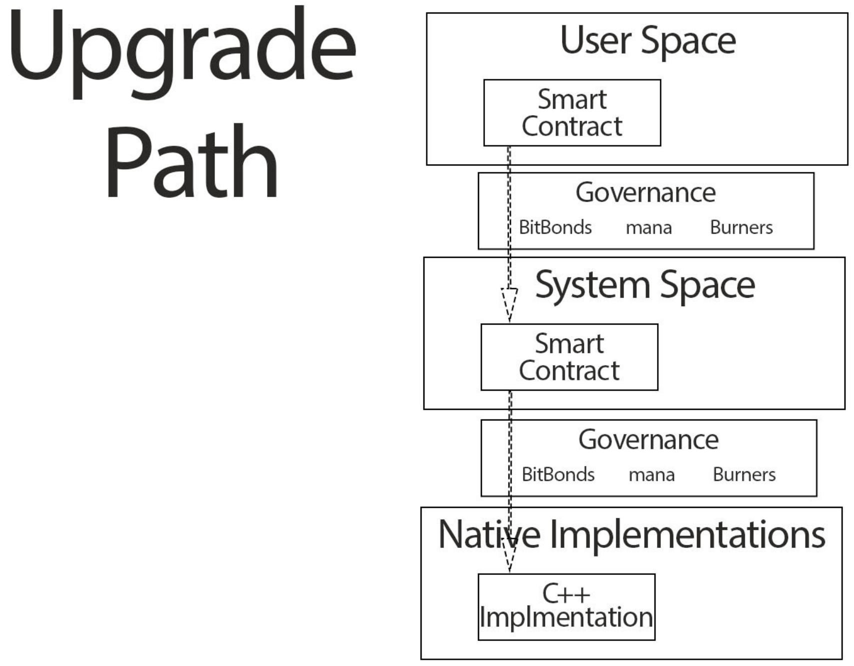
But the beauty of Koinos is that if there are additional primitives that would provide similar value, they can be run as user contracts first and then be upgraded in-band to system contracts. This enables Koinos stakeholders to add behaviors to the blockchain without having to execute a hardfork. In the meantime, a native C++ implementation can be made available and eventually hardforked in to improve performance further.

Explicit Upgrade Path

Native software upgrades are not inherently evil. They're just extremely inconvenient.

A surprising number of blockchains seem to assume their code as of the genesis block will be perfect, and never need to be upgraded with new features or bugfixes. Experience has shown this simply is not the case. As blockchain designers, we've decided to recognize this reality and future-proof Koinos by explicitly acknowledging the need for system upgrades, and designing a path to allow most upgrades to occur in a way that is completely painless for users.

Practically *every feature* on Koinos will be done through smart contracts. Unlike Ethereum and EOS, on Koinos most system behavior can be upgraded without a hardfork or software upgrade. Even critical components like the consensus algorithm, block production, resource management, digital signature algorithms, account systems, and account authorities can be upgraded without a hardfork. Governance alone will determine whether an upgrade should be made, and individual upgrades can be pushed to the network much like an operating system patch but with an on-chain record of the entire upgrade path.



Many software upgrades will only contain changes that are optimizations (replacing WebAssembly with native code) or features unrelated to the blockchain itself (such as networking-related code). Such purely technical upgrades will have very little controversy for acceptance: Every single user benefits from things like increasing the speed of system calculations, or more efficient handling of TCP/IP connections.

Where upgrades become contentious is when they change the system's economic behavior. Changes to resource limits, resource allocation, token printing, or governance mechanisms all tend to generate controversy. Such upgrades will inevitably be needed at some point, which means the controversy is also inevitable. The optimal way of handling this is to recognize it will happen, plan for it, and have an explicit on-chain process to resolve the controversy. Governance will determine whether an upgrade should be made, and on-chain records will exist to show the entire upgrade path: Proposal, ratification, and coordination of the exact switchover date.

Best of all, by including as much of the system as technically possible as system calls, the modular upgradeability of Koinos means that no technical decision is ever "final." The protocol can continue to get better and better over time while maintaining an on-chain record of all past behavior, governance decisions, upgrade semantics and switchover.

Unlike other protocols for whom time is their worst enemy, thanks to modular upgradeability, time is the strongest ally of Koinos.

The Hardfork Crisis

The evolutionary nature of Koinos requires a total shift in how one thinks about using blockchains to solve problems. Before Koinos, blockchains were designed to do all the "right" things at the moment of launch, with little attention being given to how those things would be upgraded. This design philosophy has led to a hardfork crisis. The primary bottleneck preventing blockchain adoption is not about missing features, it's about the stagnation that results when all platform upgrades must be bundled into time consuming, risky, and political hardforks. There is a reason, after all, that they are called *hard*-forks.

Before Koinos, blockchains tended to mix the concepts of feature completion, maturity, and valuation. Their goal was to achieve a mature, feature complete blockchain with a high market cap. But this is self-defeating because as value accrues, the downside risk associated with an update increases. The most successful players tend to become stagnant because the burden of upgrading gets too high once you become big.

All systems have tension between conservatism and innovation. Maturity and success create forces that steer systems away from innovation. By minimizing the burden of upgrading and creating explicit processes for upgrades to occur, a highly successful Koinos should still be able to find a balance that makes much more room for innovation compared to other similarly successful blockchains.

Purposefully Feature Incomplete

Koinos does things differently. Koinos is purposefully feature-incomplete but is equipped with a well-defined upgrade mechanism. By reducing the pain associated with upgrades, Koinos can continue to improve itself even after it has matured.

Vertical Scalability

StateDB

StateDB started as an evolution on chainbase, but has become its own beast entirely, replacing chainbase in the process. On BitShares, Steem, and EOS, the database tracks the most current state. That is the head block state plus pending transactions. This means the database never actually reflects the current state of the blockchain. In order to address that issue, when each block is applied the pending transaction state is undone, the old values are written back to the database, and then the block is applied. One problem with this approach is that most of the time this means performing the exact same calculations again and writing the same state back to the database that was just there which is inefficient.

The Ticking Time Bomb

An even bigger problem, however, is when a fork occurs. In that case a great many blocks have to be undone and reapplied, and all the related calculations along with them. People might be tempted to think of forks in the context of “hardforks” which are rare, and ideally planned, events. However, the entire purpose of a blockchain is to recognize forks and recover from them. This is what we mean by “Byzantine Fault Tolerance.” So forks are actually relatively common, and the problem with the traditional blockchain database approach is that it creates a situation where when the blockchain is near capacity, a small fork can spell disaster and force database contention to dangerous levels. This means that a blockchain might look like it is performing perfectly well for a very long time when in reality it has a ticking time bomb within it that is waiting until just the moment when people are most reliant on it to go off.

StateDB takes a fundamentally different approach to managing this database that provides several key benefits:

- 1. A new block can be applied while readers read from the state of a previous block which will make Koinos more responsive to API requests.
- 2. Multiple blocks from multiple forks can be applied in parallel without any contention on shared resources.

- 3. Fork resolution does not require writing to the database at all so Koinos will be able to run at a higher percentage of maximum capacity without suffering significant penalties from forks.
- 4. Live snapshots can be taken of Koinos because a thread can write the state at a specific block to disk in the background while the node continues processing live block.
- 5. Node failures will be minor inconveniences as opposed to the catastrophe that it can be on other blockchains.

Positive State Deltas

StateDB accomplishes this, in part, through the use of what we call, "positive state deltas" which allow Koinos to have a single backing database that tracks irreversible state and never needs to be undone. Instead of storing old values of objects, it stores new values of objects, and projects those new values across multiple undo states (state nodes in StateDB vernacular) to present the consensus state at a given block. By storing positive deltas that are then collapsed during a read to provide access to the current state, undoing and redoing state is as simple as choosing which deltas to collapse which means that switching forks does not have implicit writes in the algorithm. Furthermore, this can support processing a new block while reading state from the previous block in parallel.

In addition to the advantages listed above, this enables Koinos to minimize RAM usage in nodes by efficiently ferrying irreversible transactions out of RAM and into secondary memory. Since Koinos has a single database of irreversible state that never needs to be undone, that database can be stored entirely on disk and never brought back into RAM to resolve forks. Because StateDB is designed to work with RocksDB, performance can be maintained even while commodity storage media is used.

This focus on optimizing the blockchain to use commodity hardware has the added benefit of making node operation accessible to ordinary people, not just those with access to the enterprise-grade hardware used in most other general purpose blockchain networks. This feature alone will reduce the cost of running a Koinos node by as much as 75% compared to the most commonly used general purpose blockchains and make node operation accessible to far more people, thereby improving decentralization.

State-Paging

Note: while we have completed all of the algorithms for state-paging and validated that it works, it is more challenging to assess whether the trade off between bandwidth consumption and storage is net positive in real world conditions. This feature should therefore still be considered experimental.

State-paging takes the vertical scalability of Koinos, already beyond other major platforms, and takes it to a whole other level. The purpose of paging in an operating system is to move data that might be needed in the future, but doesn't need to be accessed immediately, to disk which is much cheaper than RAM. In the same way, state-paging takes decentralized memory management to the next level by enabling Koinos to remove unused state (what we call "ancient state") from the blockchain entirely while retaining the ability to bring it back *and cryptographically verify the data for correctness.*

Similar to how modular upgradeability separates contract behavior from implementation, state-paging separates ownership of data from the data itself. Koinos stores a kind of "digital deed" that includes the cryptographic information needed to authenticate paged-out data.

It is important to emphasize that once the data is paged back into the blockchain it is just as if it never left which means that it is every bit as trustworthy as the data stored on any other blockchain. This is what we will prove in the technical whitepaper dedicated to state-paging.

Existing general purpose blockchains establish ownership of data by storing every smart contract that has ever been uploaded to the blockchain in nodes which are optimized for decentralized computing, not low cost data storage. Koinos, on the other hand, only keeps the smart contracts in state that people are actually using, and pages out old smart contracts that no one is using.

If a developer wants their smart contracts to remain in state despite their inactivity, they will be able to do so either fee-lessly (using mana), or by paying in KOIN for the resources that smart contract will consume just as they do on Ethereum. State-paging gives the developer a powerful new option; they can store their unused smart contracts in any key/value database like Amazon S3 and only pay to bring it back into active state when a user calls it.

But the real value accrues to the network as a whole which is no longer bearing the burden of an ever increasing number of smart contracts that will never be used again! This will enable Koinos to remain "forever young."

A Strong Foundation

The Koinos blockchain framework is an entirely new blockchain architecture that was built from scratch to serve as the ultimate foundation for the Koinos mainnet. The goal was not to build a blockchain that does everything right away, but to build a blockchain that can rapidly acquire the features it needs, when it needs them, through modular upgradeability. Thanks to modular upgradeability, complex behaviors (like sharding) which are taking years to add to existing blockchains, can be added to Koinos without a hardfork. In other words, the real horizontal scaling solution is modular upgradeability.

In order to deliver an amazing developer and user experience while accommodating the rapid growth that we expect to result from modular upgradeability, we developed positive state deltas and state-paging to ensure that the Koinos network can continue to deliver high performance as it grows while ensuring that node operation remains affordable and accessible.

Testnet

In January or February 2021, we plan to launch what we refer to as an "Ethereum Equivalent" Testnet. The goal of this testnet is to test the Koinos blockchain framework and to demonstrate its capabilities, including the ability to rapidly add different features to the framework. Thanks to modular upgradeability, the features of the testnet can be totally different than mainnet. In order to expedite the delivery of the testnet and the mainnet, it will have a standard Proof of Work consensus algorithm, address-based accounts, and smart contracts, just like Ethereum. Unlike Ethereum, it will have fast block times, fee-less transactions, and WASM smart contracts. This will enable developers to begin experimenting with WASM smart contracts on Koinos and battle-testing the framework.

Mainnet

Because all of these features will be implemented as smart contracts, we can rapidly release a testnet and then shift our focus to developing the smart contracts for the mainnet. While we have mapped out all of those features and have a high degree of confidence in our ability to deliver them in 6 months (3 months after the launch of testnet), it is important to emphasize that this work has not even yet begun. That being said, here is a list of the features we plan to implement on mainnet:

1. New Consensus Algorithm
2. Mana Resource System a. Unlimited Free Account Creation b. Fee-less Transactions
3. Fungible Tokens
4. Non-Fungible Tokens a. Blockchain Bonds
5. Automated Market Maker

Economics

ERC20 economics

The Ethereum ERC20 phase of the KOIN token is programmed to mint at most 100,000,000 tokens. An emission curve of decreasing slope describes how many tokens may be minted at a given time. The tokens will be minted over 180 days. The exact number of tokens minted in the ERC20 phase depends somewhat on user behavior, but there is a hard-coded upper bound: It *must* be less than or equal to 100,000,000 tokens. This limit on minting is enforced by the smart contract. No hard lower bound on the number of minted tokens is directly enforced. However, we would be very surprised if fewer than 90,000,000 tokens were minted by the ERC-20.

Running a proof-of-work miner program will allow any Ethereum user to mint the ERC20 KOIN token based on the computational power they devote to mining. The mining process requires a small amount of ETH for transaction fees charged by the Ethereum blockchain. Since each miner can set a customized difficulty, the miner program allows the user to specify a desired mining rate (proofs per day or per week). More proofs should reduce the variance of returns, but this comes at a cost; it will submit more Ethereum transactions, which is more expensive.

Upper bounds

- At most 100,000,000 KOIN will be minted during the ERC20 phase.
- After the airdrop on the Koinos blockchain mainnet, the Koinos blockchain will mint KOIN tokens at a rate of up to 15% per year.
- Therefore, it is quite possible that up to 115,000,000 KOIN may exist one year after the launch of the Koinos mainnet.
- There is no specific long-term upper bound on the amount of KOIN that will ever exist.

Importantly, Koinos's extreme upgradeability adds some degree of uncertainty to any discussion of the system's future trajectory:

- The Koinos blockchain's governance system will be able to upgrade any part of the system.
- We recommend that governance system participants should not ratify any upgrade that results in more than 15% per year minting.
- However, it is possible that governance participants will not follow this recommendation.
- This means it is theoretically possible KOIN tokens may be printed in excess of the 15% per year limit *if such changes are approved by governance*.

ERC20 economics technical details

The mining process works by having miners present proofs to the ERC20, each of which is worth a number of hash credits determined by the difficulty of the submitted proof. For example, a difficulty of 20 bits means a particular mining attempt has a success probability of 1/2^20 and on average it requires 2^20 = 1,048,576 mining attempts to find such a proof. Therefore the proof is worth 1,048,576 hash credits based on the average case. Bear in mind, it is generally going to be the case that the user who finds the proof is either lucky or unlucky (compared to the average), so we can expect to find that it actually required either less or more than 1,048,576 actual mining attempts to find.

Hash credits are not tokens, so users cannot transfer or store them. Hash credits can only be used to obtain newly minted KOIN ERC-20 tokens. The system uses an XYK market maker calculation to determine how many hash credits get one KOIN token. Conceptually the XYK market maker has a balance of hash credits, but this system balance isn't "real" since no user can obtain the submitted hash credits. The hash credits in the market maker's balance are slowly destroyed over time via an exponential decay process.

Conversely, the KOIN tokens that might be immediately handed out to users turning in hash credits are regarded as a balance from the point of view of the market maker algorithm. This system balance isn't a "real" balance either, which means it doesn't count toward the reported ERC20 supply.

The system balances always move in opposite directions. As time passes without a user turning in hash credits, the system token balance increases, while the system hash credit balance decreases. The longer this occurs, the more tokens a user will receive for their hash credits. When a user turns in hash credits, the reverse happens: The user's newly minted tokens are deducted from system token balance, while the system hash credit balance increases due to the user's turned-in hash credits.

These forces effectively create a continuous Dutch auction mechanic. The exchange rate of tokens offered per hash credit decreases continuously over time until someone submits a proof and takes the offer, then the exchange rate increases based on the XYK algorithm's slippage.

Due to the slippage, no matter how many hash credits are submitted, only a finite number of tokens limited by the emission curve may be obtained.

Mainnet economics

Mainnet genesis will be initialized from a snapshot of the ERC20.

Since smart contracts are upgradeable, governance will necessarily have the power to make changes to token minting. Some of these changes will be key parameters that require a greater consensus level.

Initial parameters will have an upper bound of 15% token supply increase per year. Initially there will be no long-term supply limit.

The newly minted tokens will reward users for participation in subsystems that are vital to system functioning or increase system value. These subsystems include:

- Block production
- Long-term staking

Note that tokens locked in these subsystems are counted when computing the 15% upper bound on minting. So if 20,000,000 tokens are locked in long-term staking and 80,000,000 tokens are not, then the annualized limit is still 15,000,000 tokens (15% of 100,000,000).

Coming Soon

We will release more information about these features in the coming weeks and months, but we wanted to make sure that people understand that they are still very much a work in progress. One of the greatest benefits of modular upgradeability is that you don't have to develop everything all at once. We can focus on the features we know developers want and need, and defer the development of others while we gather more feedback and make the features even better.

While these are the features we believe developers want based on our past discussions, if enough developers tell us that they want something different, we will be more than happy to integrate their feedback and deliver an even better product, even if it means changing some of the features listed above.

Other teams build their blockchains as if they are infallible, we engineer our products based on the *assumption* that we are fallible. We aren't perfect. We will make mistakes. Thanks to modular upgradeability, we no longer have to live with them, or worse, pretend that bugs are features.

Initial Token Launch & Distribution

Now that we have fully informed you of what we have developed and what we intend to develop, we would like to invite you to participate in the mining of KOIN on Ethereum. It is important to reiterate that the Koinos mainnet will implement a totally new consensus algorithm and not the Proof-of-Work algorithm that will be used to mine the ERC20.

We believe that Koinos can serve as a foundation for an entirely new generation of applications that give everyone ownership of their digital selves. But in order to ensure that it can reach its full potential we want to make the launch of the mainnet as open, transparent, and decentralized as possible.

The PoW algorithm is memory-hard and GPU resistant which means that there will be little benefit to mining with anything other than a CPU, which makes mining accessible to ordinary people. In order to make mining even more accessible, we will be releasing a mining client that will make it very easy to mine KOIN on Ethereum.

PoW Algorithm

Our top priority when designing the ERC20 PoW algorithm is accessibility. We want everyone to be able to participate in the initial distribution of KOIN and not just those who can afford expensive hardware. To accomplish this we designed the algorithm to be "memory hard."

Our algorithm exploits the computer science principle that as the memory gets faster and faster, it gets smaller and smaller. To illustrate this point, the fastest memory of all is "cache" and it is so small that it exists on the CPU itself which is why accessing data that is in cache is faster than data that is in RAM. Whenever we are talking about "optimizing" a program, we are usually referring to how we take full advantage of this principle. In the case of a memory hard algorithm we are doing just the opposite. We are leveraging this principle to slow down a program that is running in an expensive computer with tons of fast memory. This evens the playing field with people who are running the same program on a less expensive computer.

The way we do this is by having the algorithm operate on a large amount of memory that is randomly accessed to circumvent the benefits of cache. An added benefit of this is that having fast memory in ASICs (machines custom-built to solve PoW algorithms) is prohibitively expensive.

Memory hard algorithms also tend to be GPU resistant. GPUs are optimized for performing the same set of instructions on different data inputs. By carefully choosing the correct memory size of the algorithm, CPU performance can be optimized over GPU performance. It has been demonstrated that CryptoNight performs competetively, if not better, on a CPU than a GPU for smaller memory input sizes [@feng-memory-hard].

Ensuring accessibility is also one of the reasons that we are choosing a 6 month mining window. Optimizing an algorithm for a GPU and/or ASICs takes time. Our algorithm is novel and will be replaced entirely with the launch of mainnet, which will use a totally different algorithm. The transient nature of the algorithm will reduce the financial incentive of developing optimized mining software. Our miner will be released open source, and we expect contributions that optimize mining. For this reason, the mining algorithm allows for multi recipient mining, so that optimizing the algorithm can be profitable for developers. We are also releasing the miner under GPLv3 to continually preserve the open community nature of the distribution of KOIN.

Generalized Birthday Problem

The PoW algorithm can be thought of much like the card game "Set" except that each card is 2mb in size and has 256 properties, so each card is unique. When mining, your computer is trying to find 10 different "cards" that match more closely than any other miner. Because every card is unique it is not possible find 10 identical cards.

Our specific algorithm is based on the generalized birthday problem. The memory input is based on a seed input which is a recent Ethereum block hash. A proof consists of a miner address, miner pow height, target difficulty, recent Ethereum block hash, recent Ethereum block height, and nonce. This data is referred to as the secure struct and is hashed to create the secured struct hash, S. We have designed a pseudo random number generator based on a polynomial with coprime coefficients along with modulo arithmetic to return an index in to the memory input. The generator takes S as an input combined and output 10 indices in to the memory input. These inputs are non-associative which leads to frequent cache misses and is the memory hard portion of the algorithm. The seed lookups are XORed together along with S to get the result of the work. The work is valid if its value is less than the target difficulty specified in the secure struct.

Because our mining algorithm is not competing for block production, we can support per miner dynamic difficulty. The mining contract uses an XYK market maker to determine the price of KOIN in hashes. When a proof is submitted, the miner is credited with hashes equal to the expected number of hashes required to create the proof and they buy KOIN from the market maker, increasing the difficulty. Over time the mining contract creates new KOIN, which buy hashes back from the market maker to reduce the difficulty. Each miner also specifies their own unique PoW height.

The combination of inputs results in an algorithm where each miner is only in competition with themselves and the only variable they need to choose is their target difficulty, which can be derived in our miner from their current hash rate and how often they wish to submit a proof. Each proof submission requires around 118,000 gas regardless of the difficulty of the proof. A miner can more efficiently mine (in terms of transaction fees) by submitting more difficult proofs less frequently. But this approach creates more variability. Variability can be reduced by choosing a lower target difficulty with the tradeoff of having to spend more on transaction fees.

Funding Development (No ICO)

As we are foregoing an ICO on the principle of a fair launch and distribution, we are asking that all miners of KOIN consider donating 5% of their tokens to OpenOrchard to help fund our existing and future development of the Koinos ecosystem. Our miner will have this feature enabled by default, but any miner can opt out if they choose.

Mainnet Airdrop

When the Koinos mainnet launches, tokens will be airdropped based entirely on the Ethereum mining, therefore the token that will be mined on Ethereum will constitute the initial token distribution of Koinos.

Once again, you can find the miner here.