EtherCore - Secure, Scalable and Decentralized Application Platform Powered by ProgPow Consensus.

V1.8.1

EtherCore is a smart contract application platform to bring back security, scalability and decentralization again for DApps. Current Blockchain applications like digital tokens and DeFi applications are deployed on one of the biggest DApp platforms like Ethereum or EOS which is considered to be centralized among the other Blockchain platforms that are being built by the active community. EtherCore is here to solve the decentralization problem for Ethereum, which will lead to the most secure and stable DApp platform for every Blockchain developer. Built by the open-source mind, EtherCore is ready to scale the network with the assist of an exclusive sidechain solution, which may keep the mainnet decentralized without any sacrifice of network performance. Via EtherCore, open source developers and miners can build a community in which they can cooperate their business with each other.

Table of Contents

Vision

Blockchain is a growing list or some set of specific records, called blocks, that are linked together using cryptography. Each block contains a cryptographic hash of the previous block, nonces, a timestamp of the creation of block, transactions, and smart contracts. Blocks are small unit of Blockchain and a collection of various transactions that are recorded on public Blockchain. Transactions are any actions that users perform on online network, whether shopping on eBay, paying bills, receiving a bounty, etc. When a transaction is formed by a user, it is sent to the so-called mempool, where it waits until it is being verified and stored to one of the blocks. Once this happens, it is no longer possible to revert or modify the transaction where we consider the transaction is secure and immutable.

Blockchain technology was introduced in 2008 with the launch of the Bitcoin currency, and since then Developers and Entrepreneurs have attempted to generalize the technology to support a wider range of applications on a single Blockchain platform. The technology of the Blockchain forces you to take a fresh look at the exchange of values, documents, money, and services. It removes intermediaries and allows users to directly interact with them. That's why Bitcoin, the first working product of Blockchain, has no doubt of being the greatest invention comparable to the discovery of the internet.

Blockchain is the best technology to make digital goods and data into the function of assets. Developers and Entrepreneurs use this technology to advance the manufacturing process and digitalize the analog assets and contracts into some form of transparent, permanent record. The data derived from public Blockchains are immensely valuable to entities that operate in multiple industry verticals. Valuable insights can be extrapolated from processed data and used to determine trading decisions, fee estimation and investment strategies.

Blockchain is being adopted by various industries. For example, Blockchain has thriven the industrial spaces from video games to supply chains. Mobile games are being adopted into Blockchain for wide use, and there are a number of efforts and industry organizations working to

employ Blockchains in supply chain logistics and supply chain management. As another example, some cloud service providers are providing Blockchain software as a form of service called "Blockchain as a Service", aka BaaS, to support the industry where the Blockchain can be adopted and utilized for the better product.

Like these solutions, EtherCore aims to be a powerful Blockchain platform that can deliver valuable insights that will be licensed as technology support. Decentralized applications deployed to multi-chain scalable solution can scale to the demands of the users of DApp. DApp is composed of smart contract and multiple public transactions that are made from users of DApp. There are floods of transactions over Blockchain platforms and current public Blockchain networks lack capacity to handle those traffics. Recent research explains that Ethereum network could be scaled over by implementing sharding in the future.

Scalable databases are sometimes consists of a horizontal partition of data witch is called database sharding. Each separate block of data is held on a separate database server instances to spread load of network capacity to horizontally scale the database. Where distributed computing is used to separate load between multiple servers (either for performance or reliability reasons), sharding approach may also be useful. However, sharding requires some technical complexity on database so it requires to be handled carefully by trained database engineers. EtherCore fulfills the requirement of sharding solution and it can be utilized via using Blockchain architecture.

Not to only mention about scalability, security is also crucial for Blockchain platforms. Digital payments like PayPal or simple payment solutions are very convenient, however, they are vulnerable for hacking attempts and some serious threats like denial-of-service attacks. To become and serve as a digital payment solution, these measurements for network security are important. That's why most of the Blockchain platforms like EOS and Ethereum is implementing secure consensus algorithm like Istanbul Byzantine Fault Tolerance consensus.

Although Hard fork can be used for software upgrades on Blockchain world, it can be a security threat for DApp ecosystem, that's why we need a stable, and secure consensus rules to prepare for safe, and sound upgrades on DApp platforms. Istanbul BFT has three different

consensus mechanisms, and the block is protected against changes, which means transactions can be final after it is mined and confirmed on the network. Blockchain can only be secure if there are confidence for transaction finality otherwise the transaction may be reverted or interrupted by unauthorized third party with bad faith.

In addition for providing mineable mainnet, EtherCore will bring new sidechains that can scale horizontally, and will adopt Istanbul BFT consensus algorithm as a foundation which is one of the Proof-of-Authority implementation Proof-of-Authority implementation. DApp ecosystem can be supported under a secure, fast, and cost-effective smart contract platform that will facilitate decentralized finance. Like MakerDAO on Ethereum platform did, decentralized finance, in short, DeFi, will be able to provide banking service to unbanked population around the world.

Currently 1.7 billion people on the globe are not able to receive any banking service. While it is still possible to access the internet, banking service aren't being provided to majority of people due to low social credit or expensive costs of transaction, lack of local infrastructure or physical security. However, DeFi solutions can make them banked beyond the limitation of local environment, whether the previous banking system is available or not, DeFi solutions make p2p transaction possible at low cost it can be.

When it comes to DeFi, EtherCore is designed with lessons of original Ethereum platform, which used to provide DeFi solutions like Bancor network or Dai stable coin. EtherCore has listened to what DApp and DeFi developers want, and that's why we've improved speed and latency at the greater limit. Every existing exchanges have rate limits on their API and it is a well-known bottleneck for DeFi service providers when it comes to collecting various datas from them. In addition to providing an advanced data analysis service, EtherCore could be a stable backbone for them, since EtherCore can scale horizontally, almost infinite requests can be handled over EtherCore node.

From inception, DeFi and DApp markets will be supported by EtherCore nodes and various infrastructure built by EtherCore developers. A secure, fast, and cost-effective web3 interface will facilitate DApp users and builders. Transactions and various contracts can be deployed at low cost it can be. Compared with other DApp solutions and platforms, we will provide competitive database solution with enhanced security coming from highly decentralized and transparent network. Every protocol upgrades can be conducted without a single delay. Mission-critical Blockchain applications including nodes and miners will definitely support your Blockchain business and users using your innovative product.

Performance Overview

Blockchain platform is the backbone of any decentralized applications. Blockchain based decentralized app developers interact the platform with special tool called node. While they use multiple programming libraries on the frontend, nodes will fetch the latest ledger information from network peers, interact with libraries from publicly exposed API endpoints, and deploy contract or contract transactions to the network. While we ensure the components like libraries or p2p connection are robust and stable, what makes the development on Blockchain unstable is the slow confirmation time of the transaction and overload of the network.

Almost every Blockchain platforms regulate transaction capacity to ensure the guaranteed quality of Blockchain, for example every Blockchain consensus have their own writeable nodes, which in case of the PoW Blockchains are mining nodes and for DPOS it is called Block Producer and every nodes must have the stable connection between them which means the input of the transactions must be limited under the number of the writeable nodes can handle. This is what we call the degree of network stability.

While we build decentralized applications on different kinds of Blockchain platforms, for example, Tether coin was meant to build on Bitcoin's 2nd chain layer solution called Omni layer. However, due to the expensive price of transferring the token on Bitcoin network and the lack of network capacity pushed Tether to issue on multiple smart contract supported chains like Ethereum, Tron, or EOS. However, the problem has not been solved, and they are still looking for opportunity to issue their token on reliable platform.

What makes the DApp platform reliable is the amount of how much DApp can hold it's users and it is being determined by several various factors of the characteristics of the Blockchain. In short, there are few known requirements for Blockchain platforms to successfully host the Blockchain applications.

What we want from Blockchain applications could be the unique function. However, every known decentralized applications are not very different from any legacy internet applications,

from gambling to issuing new tokens, legacy internet can do it better than any decentralized applications until they meet the requirements of gaining widespread use.

Let's think of building a DApp that is in competition with bigger services such as Uber, Amazon, Facebook, or eBay. Their common ground is that they offer cheap solution for their customers and their business model is to handle larger traffics than their competitors, they collect lower margin per customers however they gain bigger market shares. To fulfill their customer's demand they often develop their IT infrastructure to scale by the demand.

If those big players growth were limited by the capacity of the network, they wouldn't be able to thrive like they were in now. And if the Blockchain applications were trying to replace their position, it will likely fail since no one will use their platform without solving the problems of transaction costs and network capacity. Due to the small capacity of the Blockchain platforms, it is normal that a single DApp would occupy most of the allowed transaction capacity of the network. EOS and Ethereum might be a good example.

Before we deploy any kind of financial contracts on the Blockchain, Bitcoin used to serve as a backbone or foundation of the digital payments where cryptocurrencies are needed as an alternative to PayPal or Visa from online shops, where we use the cryptocurrencies with legacy web 1.0, it was normal to use Bitcoin for cheaper payments and faster transactions, for example, buying a coffee with Bitcoin. However, due to increased rate of transaction fee and limited capacity of network, it is almost impossible to buy a coffee with Bitcoin.

Like those examples demonstrate the problem where scalability is a necessary feature for all kinds of Blockchain applications, whether or not it supports web 3.0 or smart contracts, scalability should be considered as a primitive accomplishment and the platform could scale by the demand. Without those concerns the Blockchain platform or the decentralized applications on that platform will likely end up as a vaporware.

2.1 Scalability Challenges

The scalability of Blockchain platform will depend on how much it could scale upon the demand of the platform, and it will vary on the types of the Blockchain applications deployed on the Blockchain platform. Decentralized application developers will face the complains of the customers if the cost of using their DApp is expensive, and for companies whose shares are impacted by the profit gained by the use of platform, cost for the basic use of Blockchain platform will be in their consideration for choosing the right payment solution.

For example, at the earliest days of Bitcoin, some software sellers were accepting Bitcoin for their payments from their customers. However, due to volatility of Bitcoin and the rise of the transaction cost came as a realistic wall to accept Bitcoin as payments, and therefore the payment option was replaced to something else.

Not only a simple payment but the scalability issue also effects decentralized applications that are deployed on the Blockchain. If we were building a simple exchange for tokens such as decentralized exchanges, in short DEX, traders will need to pay more than half of the commission for transaction fee paid for sending a simple transaction to Blockchain.

Higher transaction fee will affect the customers user experience and thus Blockchain platform with lower capacity will not be preferred as a main payment solution. Now days there are plenty of competitors compared to the Bitcoin and they offer their transaction solution in lower fees. Instead of using those Blockchain platforms besides the legacy solutions like PayPal or Visa, Blockchain platforms are often considered to be more complex, higher fees, and slower confirmation speed on network.

For Blockchain applications to address these concerns, they need to understand how the fee market works, and fee market depends on the capacity of the network. While we deal with finite capacity of transaction capacity that nodes and miners can handle at the same time, Blockchain applications and smart contracts need to transact with the network, and the market will put the price on the priority of the transaction confirmation and this way we can handle the limited

capacity of the network wisely. This was first designed by Satoshi Nakamoto to prevent denial of service attacks on Blockchain so spammer couldn't throw millions of spam transactions to overload the whole network. While this method was considered as efficient choice, soon after Bitcoin become popular Worldwide, cost for a single transaction has skyrocketed and the motto of using Bitcoin for cheaper transaction cost soon became worthless.

To address this fee market problem developer thought that increasing the network capacity would increase the supply of the fee market and it will fulfill the demand of users on Blockchain network. However, due to the boundary of network capacity of Bitcoin it didn't worked and the transaction cost of Bitcoin remain high while speed of confirmation was slower than non-Blockchain payment solutions.

In order for Blockchain applications to compete with non-Blockchain payment applications those challenges for scaling must be done and there are multiple suggestions provided by protocol developers of the world. As we want the transaction capacity of the network to increase, it is common knowledge that your peer server should process more transactions than other peer servers, for example the hardware requirement is high for higher density Blockchain platforms which aims for higher transaction capacity on network, like EOS or Ripple. Of course enterprise solutions would have bigger hardware capability and network capacity to handle more transactions for real world use.

Thus, it is essential to have good support from good hardware provided by good miners, and the new consensus algorithm to pick the good miners who have good hardwares that can scale horizontally is needed. Without the utilization of consensus algorithm it would not be able to scale as demand of the decentralized applications.

2.2 Security challenge

While the support from good consensus algorithm and good miner hardwares are essential for scaling the network, the concerns for network security is also a major problem. What we could think from Bitcoin is that although it is slower than any altcoins or different Blockchain solutions, it has one of the most secure and redundant Blockchain security in the world.

The advantage of proof-of-work consensus algorithm is that it allows no one to revert or interrupt the content of Blockchain ledger. Now days there are lots of financial crime and most of them are related with counterfeit money or scams related with cash, what we could tell about Bitcoin is that if you know how to use the coin correctly, it is one of the most secure banking system when it combines with the inherent tools like multi signature wallet or cold wallet system.

It is the common knowledge that Bitcoin uses public key system and when the matching private keys are fully controlled offline, you would be able to generate the necessary information to build the transaction online and sign them offline with the matching keys, submit the raw transaction to one of the Blockchain nodes available on the network and wait for the confirmation, once the transaction is confirmed it is considered as secured since they are not able to be reverted by anyone unless the network is being centralized like the non-blockchain systems do.

However, due to the low participants of the network on other Blockchain systems, some Blockchain may suffer from 51% attack which is an exploit for the unsecured Blockchain platforms and they are being dominated by the specific 3rd party unlike Bitcoin. Multiple Blockchain developers have worked to prevent this problem and therefore there are variety of Blockchain consensus algorithms have been designed and developed to secure the alternatives.

Security of any kind of payment network is essential and it has to be done by the developer party of the platform, for example if you are opening a new website for selling goods online however you have not considered about cryptography of payment information, the information of your

clients would likely to be abused for fraud and therefore there will be no clients who wants to risk their financial information for buying goods.

Like the above security concerns, no one will want your transaction to be abused or modified by the unwilling 3rd party and thus you will need a secure payment solution to cover against malicious ones, that is why Bitcoin has developed and implemented proof-of-work consensus algorithm to combat against financial crimes and become one of the most secure money that has been ever made.

Bitcoin prevents confirmed transactions being modified or interrupted by duplicating every transaction that has been made since the beginning of the Blockchain which is called a genesis block, and it is being verified by node manually when the node receives new information about the Blockchain or what we call it sync the Blockchain, it calculates the cryptographic hashes about block headers and merkle routes.

Some people think that Blockchain is slow due to slow sync speed but it becomes slower if the Blockchain has a massive amount of transactions that has been deployed since the beginning due to it's verification process inside the node. Merkle roots of the transaction is being recalculated on every peer nodes and proves if it is the right transaction to make sure it is not being replaced by the fake ones.

Not only the sync process is special also the generation of the transaction is unique, not every nodes are participating for including transactions on the nodes. To prevent any technical fraud on the network, nodes are participating in some kind of computing power and those who have the dominant power in the network is being elected as a trusted node and they include transactions into the chunks of datas which is called a block and the whole process including calculation of the nonce and block headers is needed for proof-of-work consensus Blockchains to pick the most honest nodes in the network.

Composing EtherCore

1.1 Overview

EtherCore is a Blockchain platform to solve the trilemma issue, which is a technical limitation of Blockchain that all decentralized application platform cannot be solved at present. EtherCore will use the ProgPoW consensus algorithm to address ASIC centralization, fair mining reward distribution, and network security issues. The developers of EtherCore are developing Blockchain technology that can be applied in real life through Inter Contract Communication (ICC) and developing a platform for DApp to activate the EtherCore ecosystem.

A representative example of the technical limitations faced by the 1st and 2nd generation Blockchain is the scalability problem, which cannot significantly improve the transaction processing speed (TPS) per second as the network expands. There are two factors that are holding back in solving the scalability problem of Blockchain technology: decentralization and security. The concept that encompasses these scalability, decentralization, and security issues is the Blockchain trilemma.

The trilemma of Blockchain cannot be improved in a positive direction in terms of scalability, decentralization, and security, and it is like an equivalent exchange structure that sacrifices other elements when certain elements are being addressed. For example, to achieve the scalability of the network, a faster consensus process can be implemented by placing fewer nodes or fewer representatives than before, but as the number of nodes decreases, the degree of network decentralization and network stability decreases. In the same vein, you can achieve network decentralization by having a large number of nodes, but with reduced scalability.

To date, many DApp platforms, including Ethereum and EOS, are in the process of considering and commercializing technical solutions to solve the problem of the Blockchain, but they still sacrifice decentralization to speed up Blockchain processing. The trilemma is not getting out of trouble.

1. Addressing Trilemma issue

Our latest attempt to address the trilemma issue on a public Blockchain is to focus on decentralization and security first. Compared with other DApp platforms, while it is possible to achieve faster block times even though on PoW consensus these days (some PoW Blockchains report 5 second average block generation time with lower uncle/orphan rates which are faster than current Ethereum network which is known as 15 average confirmation time with 10% uncle rates), some may choose Delegated Proof of Stake consensus algorithm or Proof of Authority consensus to achieve faster confirmation time and scaled network capacity while they are considered to be less secure or less decentralized compared with Proof-of-Work consensus algorithm.

Still, Proof-of-Authority can be an ideal solution anyways, for decentralized exchanges they would need higher matching speed for faster trades to occur on their platform, and maybe the decentralization would likely be a second option for them. While it is not recommended to use Proof-of-Authority for real-world use case since it doesn't utilize the full advantage of Blockchain network, we may use them as a 2nd chain for scalability and the transaction can be cherry-picked to upstream PoW network via many methods including side-chain bridge or atomic swap, Inter Contract Communication (ICC) which is considered to be decentralized among those options.

Therefore, EtherCore mainnet will be built using ProgPoW Proof-of-Work consensus among with Istanbul, a Proof-of-Authority consensus sidechain to leverage the scalability for needs of specific DApps under mainnet like Tether USD, ERC-20 token or Cryptokitties smart contract. Throughout the connection with the upstream PoW mainnet, PoA sidechain can enjoy the full advantage of the decentralized network itself.

2. Analyzing ProgPoW, a Programmable Proof-of-Work Consensus Algorithm

We've developed and chose the ProgPoW algorithm as our main proof-of-work (PoW) mining and block hashing scheme. ProgPoW is a transformative algorithm of Ethash and is a proof-of-work (PoW) focused fork of the Ethash PoW algorithm. Unlike the former Ethash algorithm, ProgPoW is meant to be used for PoW-based Blockchains, such as verifying blocks. Developing cryptocurrency, the main reason that we've chosen ProgPoW is because ProgPoW is designed to be GPU-friendly and FPGA / ASIC resistant. In many years of experience in developing a proof-of-work algorithm for cryptocurrencies, ProgPoW is a superior algorithm for combating specialized mining devices. There are many PoW currencies that exist, including Bitcoin, Litecoin, Ethereum, Zcash, and much more, and they are using many different proof-of-work algorithms for computing hashes like SHA256D or Dagger-Hashimoto, Yescrypt, Aragon2D, Lyra2v, etc. When we choose hashing algorithms among them, we look for "ASIC resistance", or "FPGA resistance".

What is (or should be) meant by the term of Anti-ASIC mining is limiting the computing advantage of specialized ASIC devices. For common algorithms like SHA256 or Scrypt it is normal that ASICs or FPGA take advantage in computing hashes or mining cryptocurrencies compared with commercial hardwares. Specialized hardwares can't do other jobs, unlike commercial hardwares, they can't play games or run word processors like normal computers. Instead, they can only do mining cryptocurrencies in an effective way. They may also consume less electricity than commercial hardwares because they are utilized for mining activity with old memory chips like DDR3 to lower the manufacture costs. ProgPoW achieves some point of ASIC resistance by disadvantaging specialized hardwares, instead of utilizing the device for the algorithm, we have utilized the algorithm to benefit more on cutting edge hardwares and devices.

For example, a GeForce GTX 1080Ti card combined with high-frequency memory will achieve more hashrate power than GeForce GTX 980Ti or Nvidia Titan graphic cards that may cost more than a commercial CPU hardware and have more cores to compute but has lower memory frequency. Some pro-GPU and anti-ASIC algorithms benefits ASIC miners to establish ASIC mining

ecosystem for the cryptocurrency, although it may secure the network than GPU mining, it consumes more electricity than GPU mining ecosystem and their mining facility is centralized compared with pro-GPU mining ecosystem. Imagine a large ASIC mining facility compared with miners mining a currency with their own graphics cards instead of investing on ASIC / FPGA cards which is meant for developing mission-critical integrated circuit chips not mining cryptocurrencies and form a large factory to consume more local electricity. ProgPoW also benefits small miners to mine more coins and form a large group of the eco-friendly mining community that raises the value of cryptocurrency. Some groups of desktop GPUs or clusters of graphic cards will be enough to mine EtherCore, and you have more choice of GPU vendor to choose, buy and mine with them. However it is also impossible to estimate the number of cases of exploiting the algorithm for mining on specialized hardware, some pro-CPU or pro-GPU algorithms failed to achieve the original purpose due to some exploits, so we define here if the case happens for EtherCore / ProgPoW algorithm, we may update and patch the PoW algorithm to combat those attacks to be resistant again.

3. Adapting Proof-of-Authority Sidechain

As Proof-of-Work does, Proof-of-Authority is being used by many public Blockchain networks to secure the platform and to scale the capacity of the network for handling hundreds of transactions per second. For Ethereum, it is first used for test networks to combat denial-of-service attacks from Proof-of-Work testnet, which is likely abandoned and not secured by miners since they are less profitable for mining testnet coins.

Proof-of-Authority network is far from the public network in terms of mining, although the transactions and contracts among the network are transparent. However, transactions can't be mined and validated by miners, and the permission is very limited to specific nodes on network which is pre-defined by the founder of the Blockchain, it can be easily dominated and censored by the network creator's will like any existing web 1.0 or web 2.0 service does.

Although they are prominent pros and cons for Proof-of-Authority networks, some research of utilizing them for existing Proof-of-Work networks are on-going, and many methods to connect those chains are being developed by various individuals around the globe.

To compose the PoA sidechain network, EtherCore development team will hold a vote to elect validators who are responsible for operating and maintaining the PoA sidechain; EtherCore token holders will be able to make a vote for network governance of PoA sidechain & PoW mainnet network.

We expect that current PoA sidechain can proceed up to 1000 ~ 3000 transactions per second, PoA sidechain will be able to receive transaction demand from upstream PoW main network and will scale in parallel at the demand of decentralized applications, DApp developers will be able to deploy their own EtherCore compatible PoA sidechains and make a deposit of funds that are needed for exchanging tokens between mainnet and sidechains.

PoA network could be connected and shared by various interconnect solutions proposed by various of Blockchain developers, EtherCore development team will release the proper documentation and open-source the tools needed for composing the PoA sidechain network.

4. Addressing the distribution problem

Some well-known Blockchain platforms like Bitcoin or Ethereum is truly revolutionary, but it suffers from distribution problem which the ongoing initial distribution of Bitcoin / Ethereum is being limited to those mining tokens from the very beginning of the launch stage.

EtherCore address this distribution problem by establishing a sustainable token distribution plan. One of the key features of the supply distribution plan will be Pre-defined mining reward halving schedule and 2 giving additional incentive to the Ethereum ecosystem to attract decentralized application developers.

Unlike Ethereum, EtherCore will implement the so-called ECIP-1017 to regulate inflation caused by an undefined halving schedule in node's consensus parameters. Implementing a smooth, round reward halving schedule for the future will reduce the effect of miner's inflation. According to our model, EtherCore mainnet will have 5% ~ 9% annual inflation from Proof-of-Work reward, which is enough for securing the network. About 50% of the total supply will be allocated for PoW mining reward, which will be directly distributed to miners from the network, and 10% will be given to Ethereum holders at 1:1 ratio, to establish a good token economy for DApp development. Another 10% of the total supply will be used for Token sales, which will establish the initial price of the coin.

Max supply will be 1 billion at the end, and initial supply will be 500 million tokens, there will be some lock-up funds for private investors and development team, and we expect those tokens will be distributed to market after some kind of core development happens, to reduce the effect of inflation there will be a few years of token lock-up term for airdrop & pre-allocated funds.

5. Building a DApp economy

To support active DApp development on EtherCore mainnet, EtherCore will adopt the unique bounty system called DApp accelerator. Pursuant to a Blockchain-based on EtherCore, EtherCore holders can elect a number of DApp accelerators designed to benefit the community. The system contracts that implement DApp accelerators may not be in place at the initial launch, but the funding mechanism will. It will begin to accumulate funds at the same time block producer awards start. Since the DApp Accelerator System will be implemented in the mainnet, it can be added at a later date without a fork.

To make the funding possible, EtherCore developers will add the pre-allocation of funds and not only premine, but some amount of funds can also be collected by some donations from stakeholders of the token to increase the value of the coin. The community can develop the DApp, register for vote, and be financially supported.

1.2 Specification of ProgPoW

ProgPoW is a new Proof-of-Work algorithm which is an extension of Ethash to combat specialized hardwares for mining cryptocurrency. Although Ethash was considered to prevent ASIC (Application Specific Integrated Circuit) chips from mining cryptocurrencies, several exploits from algorithm allowed chip designing companies and some bigger miners to take over the network. To defeat the centralization of network and to make the network more secure, the modification of algorithm was needed and therefore we suggest ProgPoW as a successor of Ethash algorithm.

To be specific for our goal for developing ProgPoW, ever since the first Bitcoin mining ASIC was released, many new Proof-of-Work algorithms have been created with the intention of being "ASIC-resistant". The goal of "ASIC-resistance" is to resist the centralization of PoW mining power such that these coins couldn't be so easily manipulated by a few mining companies.

The development goal of ProgPoW is to utilize the algorithm's use for what is available on commodity hardwares where every miner can buy worldwide. To fulfill it's target, ProgPoW will use every computing resource available on graphic card. Compared with this, if the algorithm is to be implemented on a specialized hardwares like customized FPGA cards or ASIC boards, there should be little opportunity for efficiency gains compared to a commodity graphic cards.

In many years of experience in developing a proof-of-work algorithm for cryptocurrency algorithms, ProgPoW is a superior algorithm for combating specialized mining devices. There are many PoW currencies that exist, including Bitcoin, Litecoin, Ethereum, Zcash, and much more, and they are using many different proof-of-work algorithms for computing hashes like SHA256D, Dagger-Hashimoto, Yescrypt, Aragon2D, Lyra2v, etc. When we choose hashing algorithms among them, we look for "ASIC resistance", or "FPGA resistance".

By following the design, ProgPoW can be considered as a special extension or derivative algorithm from Dagger-Hashimoto, may parts of the code are inherited from the original algorithm and it actually relies on to make the algorithm memory intensive and thus to be ASIC resistant. While the inherited part of the algorithm will utilize the DRAM part of the graphic card,

it is possible for ProgPoW's parameter calculated by number of blocks can utilize the core of GPU. Like any other core intensive algorithm, ProgPoW will utilize the computation of graphic cards to maximize the efficiency of blocking the development of specialized hardwares.

Therefore, the major modifications from Ethash will be the following:

- Increases mix state.
- Increases the DRAM read from 128 bytes to 256 bytes.
- Changes Keccak_f1600 (with 64-bit words) to Keccak_f800 (with 32-bit words) to reduce impact on total power
- Adds a random sequence of math in the main loop.
- Adds reads from a small, low-latency cache that supports random addresses.

While a custom ASIC to implement this algorithm is still possible, the efficiency gains available are minimal. The majority of a commodity GPU is required to support the above elements. Our design goal of ProgPoW is to have about 1.1 ~ 1.2x efficiency gains if specialized hardwares were designed for ProgPoW while this is much less gain expected from 2x for Ethash algorithm or 100x gain for Equihash ASIC miners.

1. Reasons to design Commodity hardware friendly Proof-of-Work algorithm

According to the recent report, more than half of the computing power for Bitcoin network is coming from a single province area in China. If the local provider fails to provide their service, Bitcoin may face uncomfortable delays for processing payments online. Not only the downtime, but also there are a serious risk of network centralization which exposes Bitcoin to the risk of third party interference and censorship, etc.

With the growth of large mining pools, the control of hashing power has been delegated to the top few pools to provide a steadier economic return for small miners. While some have made the argument that large centralized pools defeats the purpose of "ASIC resistance," it's important to note that ASIC based coins are even more centralized for several reasons.

1. High barrier to entry: ASIC miners are rich enough to invest capital and ecological resources on a new coin miner chip. Thus, initial coin distribution through ASIC mining will be very limited causing centralized economic bias for the coin ecosystem.
2. Delegated centralization vs implementation centralization: While mining pool centralization is delegated, hardware monoculture is not: only the limited buyers of this hardware can participate so there isn't even the possibility of divesting control on short notice.
3. No natural distribution: There isn't an economic purpose for ultra-specialized hardware outside of mining and thus no reason for most people to have it.
4. No reserve group: Thus, there's no reserve pool of hardware or reserve pool of interested parties to jump in when coin price is volatile and attractive for manipulation.
5. No obvious decentralization of control even with decentralized mining: Once large custom ASIC makers get into the game, designing back-doored hardware is trivial. ASIC makers have no incentive to be transparent or fair in market participation.

While some coins and Blockchains may target individual miners for the concept of "ASIC resistance", the entire idea may be a fallacy. CPUs and GPUs are themselves ASICs. Any algorithm that can run on a commodity ASIC (CPU or GPU) by definition can have a customized ASIC created for it with slightly less functionality. Some algorithms are intentionally made to be "ASIC friendly" - where an ASIC implementation is drastically more efficient than the same algorithm running on general purpose hardware. The protection that this offers when the coin is unknown also makes it an attractive target for a dedicated mining ASIC company as soon as it becomes useful.

Therefore, ASIC resistance is: the efficiency difference of specialized hardware versus hardware that has a wider adoption and applicability. A smaller efficiency difference between custom vs general hardware mean higher resistance and a better algorithm. This efficiency difference is the proper metric to use when comparing the quality of PoW algorithms. Efficiency could mean absolute

performance, performance per watt, or performance per dollar - they are all highly correlated. If a single entity creates and controls an ASIC that is drastically more efficient, they can gain 51% of the network hash rate and possibly stage an attack.

2. Technical implementation of ProgPoW

As ProgPoW follows the design of the original Ethash Proof-of-Work algorithm but tunes drastically by parameters and tunes applied for ASIC resistance, some core features are being inherited such as DAG cache verification and SHA3 hashes applied to the algorithm.

Following is the example code for DAG cache generation which is being changed every 30,000 blocks.

```
void ethash_calculate_dag_item(
        node* const ret,
        uint32_t node_index,
        ethash_light_t const light
)
{
        uint32_t num_parent_nodes = (uint32_t) (light->cache_size / sizeof(node));
        node const* cache_nodes = (node const *) light->cache;
        node const* init = &cache_nodes[node_index % num_parent_nodes];
        memcpy(ret, init, sizeof(node));
        ret->words[0] ^= node_index;
        SHA3_512(ret->bytes, ret->bytes, sizeof(node));
        __m128i const fnv_prime = _mm_set1_epi32(FNV_PRIME);
        __m128i xmm0 = ret->xmm[0];
        __m128i xmm1 = ret->xmm[1];
        __m128i xmm2 = ret->xmm[2];
        __m128i xmm3 = ret->xmm[3];
```


```
		for (uint32_t i = 0; i != ETHASH_DATASET_PARENTS; ++i) {
			uint32_t parent_index = fnv_hash(node_index ^ i, ret->words[i %
NODE_WORDS]) % num_parent_nodes;
			node const *parent = &cache_nodes[parent_index];

			{
				xmm0 = _mm_mullo_epi32(xmm0, fnv_prime);
				xmm1 = _mm_mullo_epi32(xmm1, fnv_prime);
				xmm2 = _mm_mullo_epi32(xmm2, fnv_prime);
				xmm3 = _mm_mullo_epi32(xmm3, fnv_prime);
				xmm0 = _mm_xor_si128(xmm0, parent->xmm[0]);
				xmm1 = _mm_xor_si128(xmm1, parent->xmm[1]);
				xmm2 = _mm_xor_si128(xmm2, parent->xmm[2]);
				xmm3 = _mm_xor_si128(xmm3, parent->xmm[3]);

				// have to write to ret as values are used to compute index
				ret->xmm[0] = xmm0;
				ret->xmm[1] = xmm1;
				ret->xmm[2] = xmm2;
				ret->xmm[3] = xmm3;
			}
		}
		SHA3_512(ret->bytes, ret->bytes, sizeof(node));
}
```

SHA3 can be used by following example

```
def sha3(x):
    if isinstance(x, (int, long)):
        x = encode_int(x)
    return decode_int(utils.sha3(x))
```


```
def dbl_sha3(x):
    if isinstance(x, (int, long)):
        x = encode_int(x)
    return decode_int(utils.sha3(utils.sha3(x)))
```

ProgPoW has the following parameters for its design, The proposed settings have been tuned for a range of existing, commodity GPUs:

- `PROGPOW_PERIOD`: Number of blocks before changing the random program
- `PROGPOW_LANES`: The number of parallel lanes that coordinate to calculate a single hash instance
- `PROGPOW_REGS`: The register file usage size
- `PROGPOW_DAG_LOADS`: Number of uint32 loads from the DAG per lane
- `PROGPOW_CACHE_BYTES`: The size of the cache
- `PROGPOW_CNT_DAG`: The number of DAG accesses, defined as the outer loop of the algorithm (64 is the same as ethash)
- `PROGPOW_CNT_CACHE`: The number of cache accesses per loop
- `PROGPOW_CNT_MATH`: The number of math operations per loop

Following is the suggest parameter values for implementation.

Parameter	ProgPoW
`PROGPOW_PERIOD`	50
`PROGPOW_LANES`	16
`PROGPOW_REGS`	32
`PROGPOW_DAG_LOADS`	4
`PROGPOW_CNT_DAG`	64
`PROGPOW_CNT_CACHE`	12

PROGPOW_CACHE_BYTES	16x1024
PROGPOW_CNT_MATH	20

To achieve more performance and to give more competitive advantage of performance per watt to AMD graphic cards compared with Nvidia graphic cards, following parameter values were proposed instead and it is likely to be used for unique value for EtherCore mainnet.

Parameter	ProgPoW
PROGPOW_PERIOD	10
PROGPOW_LANES	16
PROGPOW_REGS	32
PROGPOW_DAG_LOADS	4
PROGPOW_CNT_DAG	64
PROGPOW_CNT_CACHE	11
PROGPOW_CACHE_BYTES	16x1024
PROGPOW_CNT_MATH	18

Some values were decreased to give more performance advantages for AMD cards, however, it doesn't affect the ASIC resistance at all.

By reducing PROGPOW_PERIOD, the hash rate of every period will naturally be slightly different (+/- 5%) due to the different random sequences generated. Reducing the period from ~10 minutes to ~2 minutes prevents the overall difficulty from drifting significantly in response to an individual period. As an added bonus this makes FPGAs even more impractical.

Typically compiling the kernel takes <1 second, even on a relatively slow machine. In the case of a

series of quick blocks and a system with an exceptionally slow CPU compiling for a few seconds every 10 blocks should not be a problem. This will especially be true once the planned Ethminer optimization that compiles period N+1's kernel while period N is executing is in place.

By reducing the general computation requirements of ProgPoW, after testing on a wider variety of GPUs we've discovered the current parameters unintentionally causes some AMD GPUs to be compute limited instead of memory limited. Reducing the random cache and math counts by 10% increases the hash rate on those AMD GPUs. This has no effect on the hash rate of other AMD GPUs or any the Nvidia GPUs we tested.

This is the lowest the counts can be reduced while keeping the important property of (PROGPOW_DAG_LOADS(4) + PROGPOW_CNT_CACHE(11) + PROGPOW_CNT_MATH(18) > PROGPOW_REGS(32)). This property ensures that every mix[] destination gets written to.

The random program changes every PROGPOW_PERIOD blocks to ensure the hardware executing the algorithm is fully programmable. If the program only changed every DAG epoch (roughly 5 days) certain miners could have time to develop hand-optimized versions of the random sequence, giving them an undue advantage.

All numerics are computed using unsigned 32 bit integers. Any overflows are trimmed off before proceeding to the next computation. Languages that use numerics not fixed to bit lengths (such as Python and JavaScript) or that only use signed integers (such as Java) will need to keep their languages' quirks in mind. The extensive use of 32 bit data values aligns with modern GPUs internal data architectures.

ProgPoW algorithm uses a 32-bit variant of FNV1a for merging data. The one that is being used for Ethash algorithm which is FNV1 is in a deprecated state and it is the known part of weakness for ASIC development exploit, so FNV1a may provide better distribution properties.

```
const uint32_t FNV_PRIME = 0x1000193;
const uint32_t FNV_OFFSET_BASIS = 0x811c9dc5;

uint32_t fnv1a(uint32_t h, uint32_t d)
{
    return (h ^ d) * FNV_PRIME;
}
```

ProgPow uses KISS99 for random number generation. This is the simplest (fewest instruction) random generator that passes the TestU01 statistical test suite. A more complex random number generator like Mersenne Twister can be efficiently implemented on a specialized ASIC, providing an opportunity for efficiency gains.

```
typedef struct {
    uint32_t z, w, jsr, jcong;
} kiss99_t;

// KISS99 is simple, fast, and passes the TestU01 suite
// https://en.wikipedia.org/wiki/KISS_(algorithm)
// http://www.cse.yorku.ca/~oz/marsaglia-rng.html
uint32_t kiss99(kiss99_t &st)
{
    st.z = 36969 * (st.z & 65535) + (st.z >> 16);
    st.w = 18000 * (st.w & 65535) + (st.w >> 16);
    uint32_t MWC = ((st.z << 16) + st.w);
    st.jsr ^= (st.jsr << 17);
    st.jsr ^= (st.jsr >> 13);
    st.jsr ^= (st.jsr << 5);
    st.jcong = 69069 * st.jcong + 1234567;
    return ((MWC^st.jcong) + st.jsr);
}
```


The fill_mix function populates an array of int32 values used by each lane in the hash calculations.

```
void fill_mix(
    uint64_t hash_seed,
    uint32_t lane_id,
    uint32_t mix[PROGPOW_REGS]
)
{
    // Use FNV to expand the per-warp seed to per-lane
    // Use KISS to expand the per-lane seed to fill mix
    kiss99_t st;
    st.z = fnv1a(FNV_OFFSET_BASIS, seed);
    st.w = fnv1a(st.z, seed >> 32);
    st.jsr = fnv1a(st.w, lane_id);
    st.jcong = fnv1a(st.jsr, lane_id);
    for (int i = 0; i < PROGPOW_REGS; i++)
            mix[i] = kiss99(st);
}
```

Like Ethash SHA3 is used to seed the sequence per-nonce and to produce the final result. The Keccak-f800 variant is used as the 32-bit word size matches the native word size of modern GPUs. The implementation is a variant of SHAKE with width=800, bitrate=576, capacity=224, output=256, and no padding. The result of Keccak is treated as a 256-bit big-endian number - that is result byte 0 is the MSB of the value.

As with Ethash the input and output of the Keccak function are fixed and relatively small. This means only a single "absorb" and "squeeze" phase are required. For a pseudo-code implementation of the Keccak_f800_round function see the Round[b](A,RC) function in the "Pseudo-code description of the permutations" section of the official Keccak specs.

```
hash32_t keccak_f800_progpow(hash32_t header, uint64_t seed, hash32_t digest)
{
    uint32_t st[25];

    // Initialization
    for (int i = 0; i < 25; i++)
        st[i] = 0;

    // Absorb phase for fixed 18 words of input
    for (int i = 0; i < 8; i++)
        st[i] = header.uint32s[i];
    st[8] = seed;
    st[9] = seed >> 32;
    for (int i = 0; i < 8; i++)
        st[10+i] = digest.uint32s[i];

    // keccak_f800 call for the single absorb pass
    for (int r = 0; r < 22; r++)
        keccak_f800_round(st, r);

    // Squeeze phase for fixed 8 words of output
    hash32_t ret;
    for (int i=0; i<8; i++)
        ret.uint32s[i] = st[i];

    return ret;
}
```

The inner loop uses FNV and KISS99 to generate a random sequence from the prog_seed. This random sequence determines which mix state is accessed and what random math is performed.

Since the prog_seed changes only once per PROGPOW_PERIOD it is expected that while mining progPowLoop will be evaluated on the CPU to generate source code for that period's sequence.

The source code will be compiled on the CPU before running on the GPU.

```
kiss99_t progPowInit(uint64_t prog_seed, int mix_seq_dst[PROGPOW_REGS], int
mix_seq_src[PROGPOW_REGS])
{
	kiss99_t prog_rnd;
	prog_rnd.z = fnv1a(FNV_OFFSET_BASIS, prog_seed);
	prog_rnd.w = fnv1a(prog_rnd.z, prog_seed >> 32);
	prog_rnd.jsr = fnv1a(prog_rnd.w, prog_seed);
	prog_rnd.jcong = fnv1a(prog_rnd.jsr, prog_seed >> 32);
	// Create a random sequence of mix destinations for merge() and mix sources for cache reads
	// guarantees every destination merged once
	// guarantees no duplicate cache reads, which could be optimized away
	// Uses Fisher-Yates shuffle
	for (int i = 0; i < PROGPOW_REGS; i++)
	{
		mix_seq_dst[i] = i;
		mix_seq_src[i] = i;
	}
	for (int i = PROGPOW_REGS - 1; i > 0; i--)
	{
		int j;
		j = kiss99(prog_rnd) % (i + 1);
		swap(mix_seq_dst[i], mix_seq_dst[j]);
		j = kiss99(prog_rnd) % (i + 1);
		swap(mix_seq_src[i], mix_seq_src[j]);
	}
	return prog_rnd;
}
```

The math operations that merges values into the mix data are ones chosen to maintain entropy.

```
// Merge new data from b into the value in a
// Assuming A has high entropy only do ops that retain entropy
// even if B is low entropy
// (IE don't do A&B)
uint32_t merge(uint32_t a, uint32_t b, uint32_t r)
{
    switch (r % 4)
    {
    case 0: return (a * 33) + b;
    case 1: return (a ^ b) * 33;
    // prevent rotate by 0 which is a NOP
    case 2: return ROTL32(a, ((r >> 16) % 31) + 1) ^ b;
    case 3: return ROTR32(a, ((r >> 16) % 31) + 1) ^ b;
    }
}
```

The math operations chosen for the random math are ones that are easy to implement in CUDA and OpenCL, the two main programming languages for commodity GPUs. The mul_hi, min, clz, and popcount functions match the corresponding OpenCL functions. ROTL32 matches the OpenCL rotate function. ROTR32 is rotate right, which is equivalent to rotate(i, 32-v).

```
// Random math between two input values
uint32_t math(uint32_t a, uint32_t b, uint32_t r)
{
    switch (r % 11)
    {
    case 0: return a + b;
    case 1: return a * b;
    case 2: return mul_hi(a, b);
    case 3: return min(a, b);
    case 4: return ROTL32(a, b);
    case 5: return ROTR32(a, b);
    case 6: return a & b;
```

```
    case 7: return a | b;
    case 8: return a ^ b;
    case 9: return clz(a) + clz(b);
    case 10: return popcount(a) + popcount(b);
    }
}
```

The flow of the inner loop is:

Lane (loop % LANES) is chosen as the leader for that loop iteration

The leader's mix[0] value modulo the number of 256-byte DAG entries is used to select where to read from the full DAG

Each lane reads DAG_LOADS sequential words, using (lane ^ loop) % LANES as the starting offset within the entry.

The random sequence of math and cache accesses is performed

The DAG data read at the start of the loop is merged at the end of the loop

prog_seed and loop come from the outer loop, corresponding to the current program seed (which is block_number/PROGPOW_PERIOD) and the loop iteration number. mix is the state array, initially filled by fill_mix. dag is the bytes of the Ethash DAG grouped into 32 bit unsigned ints in litte-endian format. On little-endian architectures this is just a normal int32 pointer to the existing DAG.

DAG_BYTES is set to the number of bytes in the current DAG, which is generated identically to the existing Ethash algorithm.

```
void progPowLoop(
    const uint64_t prog_seed,
    const uint32_t loop,
```

```
    uint32_t mix[PROGPOW_LANES][PROGPOW_REGS],
    const uint32_t *dag)
{
    // dag_entry holds the 256 bytes of data loaded from the DAG
    uint32_t dag_entry[PROGPOW_LANES][PROGPOW_DAG_LOADS];
    // On each loop iteration rotate which lane is the source of the DAG address.
    // The source lane's mix[0] value is used to ensure the last loop's DAG data feeds into this loop's address.
    // dag_addr_base is which 256-byte entry within the DAG will be accessed
    uint32_t dag_addr_base = mix[loop%PROGPOW_LANES][0] %
        (DAG_BYTES / (PROGPOW_LANES*PROGPOW_DAG_LOADS*sizeof(uint32_t)));
    for (int l = 0; l < PROGPOW_LANES; l++)
    {
        // Lanes access DAG_LOADS sequential words from the dag entry
        // Shuffle which portion of the entry each lane accesses each iteration by XORing lane and loop.
        // This prevents multi-chip ASICs from each storing just a portion of the DAG
        size_t dag_addr_lane = dag_addr_base * PROGPOW_LANES + (l ^ loop) % PROGPOW_LANES;
        for (int i = 0; i < PROGPOW_DAG_LOADS; i++)
            dag_entry[l][i] = dag[dag_addr_lane * PROGPOW_DAG_LOADS + i];
    }

    // Initialize the program seed and sequences
    // When mining these are evaluated on the CPU and compiled away
    int mix_seq_dst[PROGPOW_REGS];
    int mix_seq_src[PROGPOW_REGS];
    int mix_seq_dst_cnt = 0;
    int mix_seq_src_cnt = 0;
    kiss99_t prog_rnd = progPowInit(prog_seed, mix_seq_dst, mix_seq_src);

    int max_i = max(PROGPOW_CNT_CACHE, PROGPOW_CNT_MATH);
    for (int i = 0; i < max_i; i++)
    {
```


```
        if (i < PROGPOW_CNT_CACHE)
        {
            // Cached memory access
            // lanes access random 32-bit locations within the first portion of the DAG
            int src = mix_seq_src[(mix_seq_src_cnt++)%PROGPOW_REGS];
            int dst = mix_seq_dst[(mix_seq_dst_cnt++)%PROGPOW_REGS];
            int sel = kiss99(prog_rnd);
            for (int l = 0; l < PROGPOW_LANES; l++)
            {
                uint32_t offset = mix[l][src] % (PROGPOW_CACHE_BYTES/sizeof(uint32_t));
                mix[l][dst] = merge(mix[l][dst], dag[offset], sel);
            }
        }
        if (i < PROGPOW_CNT_MATH)
        {
            // Random Math
            // Generate 2 unique sources
            int src_rnd = kiss99(prog_rnd) % (PROGPOW_REGS * (PROGPOW_REGS-1));
            int src1 = src_rnd % PROGPOW_REGS; // 0 <= src1 < PROGPOW_REGS
            int src2 = src_rnd / PROGPOW_REGS; // 0 <= src2 < PROGPOW_REGS - 1
            if (src2 >= src1) ++src2; // src2 is now any reg other than src1
            int sel1 = kiss99(prog_rnd);
            int dst  = mix_seq_dst[(mix_seq_dst_cnt++)%PROGPOW_REGS];
            int sel2 = kiss99(prog_rnd);
            for (int l = 0; l < PROGPOW_LANES; l++)
            {
                uint32_t data = math(mix[l][src1], mix[l][src2], sel1);
                mix[l][dst] = merge(mix[l][dst], data, sel2);
            }
        }
    }
    // Consume the global load data at the very end of the loop to allow full latency hiding
    // Always merge into mix[0] to feed the offset calculation
    for (int i = 0; i < PROGPOW_DAG_LOADS; i++)
```

```
    {
        int dst = (i==0) ? 0 : mix_seq_dst[(mix_seq_dst_cnt++)%PROGPOW_REGS];
        int sel = kiss99(prog_rnd);
        for (int l = 0; l < PROGPOW_LANES; l++)
            mix[l][dst] = merge(mix[l][dst], dag_entry[l][i], sel);
    }
}
```

The flow of the overall algorithm is:

A Keccak hash of the header + nonce to create a seed

Use the seed to generate initial mix data

Loop multiple times, each time hashing random loads and random math into the mix data

Hash all the mix data into a single 256-bit value

A final Keccak hash is computed

When mining this final value is compared against a hash32_t target

```
hash32_t progPowHash(
    const uint64_t prog_seed, // value is (block_number/PROGPOW_PERIOD)
    const uint64_t nonce,
    const hash32_t header,
    const uint32_t *dag // gigabyte DAG located in framebuffer - the first portion gets cached
)
{
    uint32_t mix[PROGPOW_LANES][PROGPOW_REGS];
    hash32_t digest;
    for (int i = 0; i < 8; i++)
        digest.uint32s[i] = 0;
```


```
    // keccak(header..nonce)
    hash32_t seed_256 = keccak_f800_progpow(header, nonce, digest);
    // endian swap so byte 0 of the hash is the MSB of the value
    uint64_t seed = bswap(seed_256[0]) << 32 | bswap(seed_256[1]);

    // initialize mix for all lanes
    for (int l = 0; l < PROGPOW_LANES; l++)
        fill_mix(seed, l, mix[l]);

    // execute the randomly generated inner loop
    for (int i = 0; i < PROGPOW_CNT_DAG; i++)
        progPowLoop(prog_seed, i, mix, dag);

    // Reduce mix data to a per-lane 32-bit digest
    uint32_t digest_lane[PROGPOW_LANES];
    for (int l = 0; l < PROGPOW_LANES; l++)
    {
        digest_lane[l] = FNV_OFFSET_BASIS
        for (int i = 0; i < PROGPOW_REGS; i++)
            digest_lane[l] = fnv1a(digest_lane[l], mix[l][i]);
    }
    // Reduce all lanes to a single 256-bit digest
    for (int i = 0; i < 8; i++)
        digest.uint32s[i] = FNV_OFFSET_BASIS;
    for (int l = 0; l < PROGPOW_LANES; l++)
        digest.uint32s[l%8] = fnv1a(digest.uint32s[l%8], digest_lane[l])

    // keccak(header .. keccak(header..nonce) .. digest);
    keccak_f800_progpow(header, seed, digest);
}
```


3. Comparison with other Proof-of-Work algorithms

While ProgPoW is meant to be superior from gaining performance for commodity hardwares such as graphic cards, we have identified several poW algorithms performance from developing mining specialized hardwares, these are the common assumptions made by the design of other algorithms.

SHA256

- Potential ASIC efficiency gain ~ 1000X

The SHA algorithm is a sequence of simple math operations - additions, logical ops, and rotates.

To process a single op on a CPU or GPU requires fetching and decoding an instruction, reading data from a register file, executing the instruction, and then writing the result back to a register file. This takes significant time and power.

A single op implemented in an ASIC takes a handful of transistors and wires. This means every individual op takes negligible power, area, or time. A hashing core is built by laying out the sequence of required ops.

The hashing core can execute the required sequence of ops in much less time, and using less power or area, than doing the same sequence on a CPU or GPU. A Bitcoin ASIC consists of a number of identical hashing cores and some minimal off-chip communication.

Equihash

- Potential ASIC efficiency gain ~ 100X

The ~150mb of state is large but possible on an ASIC. The binning, sorting, and comparing of bit strings could be implemented on an ASIC at extremely high speed.

CryptoNight

- Potential ASIC efficiency gain ~ 50X

Compared to Scrypt, CryptoNight does much less compute and requires a full 2mb of scratch pad (there is no known Time/Memory Tradeoff attack). The large scratch pad will dominate the ASIC

implementation and limit the number of hashing cores, limiting the absolute performance of the ASIC. An ASIC will consist almost entirely of just on-die SRAM.

Ethash

- Potential ASIC efficiency gain ~ 2X

Ethash requires external memory due to the large size of the DAG. However that is all that it requires - there is minimal compute that is done on the result loaded from memory. As a result a custom ASIC could remove most of the complexity, and power, of a GPU and be just a memory interface connected to a small compute engine.

2. Conclusion

ProgPoW utilizes almost all parts of a commodity GPU, excluding:

- The graphics pipeline (displays, geometry engines, texturing, etc);
- Floating point math.

Making use of either of these would have significant portability issues between commodity hardware vendors, and across programming languages.

Since the GPU is almost fully utilized, there's little opportunity for specialized ASICs to gain efficiency. Removing both the graphics pipeline and floating point math could provide up to 1.2x gains in efficiency, compared to the 2x gains possible in Ethash, and 50x gains possible for CryptoNight.

1.3 Zero transaction fee smart contract platform

Supporting zero transaction fee has been an controversial subject due to the possibility of Denial-of-Service attack caused by a large amount of transactions, however it is technically possible to support the zero transaction fee smart contract platform in fact, it is also possible to send and include zero fee transactions in Ethereum mainnet at the small chance.

1. Potential of zero fee contract platform

By supporting zero fee transactions it will not only benefit the platform itself but also benefit foreign tokens deployed over the mainnet, for example ERC-20 tokens deployed over EtherCore mainnet will be able to provide holders to send and receive their tokens without paying EtherCore tokens.

Enabling foreign tokens and DApps to interact with our mainnet will be a chance to enlarge the EtherCore ecosystem and give them an ability to use the full advantage of Blockchain platform at zero cost.

While it is possible for an attacker to deploy multiple zero fee transactions to spam over the network, a proper limitation will be able to block the attempt of spamming over network and properly provide lower cost transactions to users. Limitation of block capacity determined by the miners of block and limitation of transaction pool size, in other words mempool or txpool, will be able to limit the capacity of spamming the network and protect against those improper attempts.

Governance Model

Technically, EtherCore will be consisted in two major parts, Proof-of-Work mainnet, and Proof-of-Authority sidechain. Proof-of-Work mainnet aims to be a decentralized hosting platform for different applications, and Proof-of-Authority sidechain aims to provide enough volatility and performance for applications deployed on the mainnet.

While it is prominent that those networks are being assisted by their code itself, we thought that a valid governance model for platform will make the ecosystem thrive and useful for new developers joining from different platforms. Giving the proper method to deploy or develop the contract will dramatically increase the productivity for them.

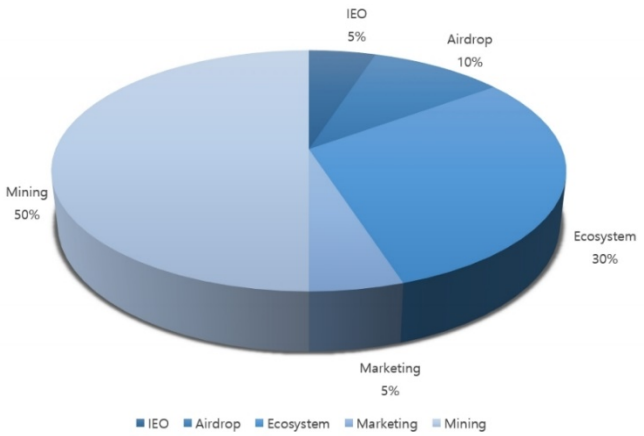
To ensure that EtherCore keeps up with the changing needs of the community, Ecosystem formation fund was introduced in the initial design of the software. This would allow for an allocation of funds to be reserved that could incentivize developers to work on community needs, critical patches, and forward-looking upgrades. This allows the system to fully utilize the strengths of a decentralized community going forward, reducing the reliance on a single source for these new additions. EtherCore has given the world a great starting point for the tool, and the ecosystem formation fund will allow the community to nurture and grow that tool.

Fund will be allocated to the EtherCore governance council and it will be paid to whom contributes to the EtherCore ecosystem by building helpful DApps and making a persistent contribution to the ecosystem. EtherCore governance council will be nominated by the stake holders of EtherCore and the council will be able to use the delegated power from the community.

Token Model

The EtherCore token (ERE) is a universal currency based on the EtherCore Protocol, distributed to incentivize different participants in the ecosystem of the EtherCore mainnet. Unlike any other different tokens, it will be issued on the dedicated mainnet developed by the team without any form of tokens distributed on the different mainnets like ERC20 on Ethereum or etc.

A total supply of 1,000,000,000 (One billion) ERE tokens will be issued during mining period and the premine for airdrop, ecosystem distribution, development and marketing funds.



- 5% of the total supply will be issued from premine pool and will be used for IEO (Initial Exchange Offering) sale to invite more participants to the ecosystem
- 10% of the total supply will be issued from premine pool and will be distributed as an airdrop to Ethereum stake holders for promotion.
- 30% of the total supply will be issued from the premine pool and will be allocated for the ecosystem formation which will be locked into the smart contract for 2 years from the genesis block.
- 5% of the total supply will be issued from the premine pool and will be allocated for the development team for marketing purpose
- 50% of the total supply will be allocated for public mining, miners who mine the blocks will receive the mining reward for the long term

Mining rewards will be cut in half after the specific period of time, to ensure long term mining for EtherCore mainnet. Unlike Ethereum, there will be a maximum cap of 1,000,000,000 (One billion) tokens for issuance.

Roadmap

Scientists and engineers on EtherCore team have created an extensive roadmap that lays out key developmental milestones for the next two years.

2019 Q1

- Initial development of EtherCore whitepaper
- Development of ProgPoW softwares

2019 Q3

- Development of EtherCore Explorer
- Development of EtherCore mining pool

2019 Q4

- Development of EtherCore Wallet
- Design the user interface for EtherCore community

2020 Q1

- Build EtherCore community
- EtherCore node distribution
- Design of sidechain starts

2020 Q2

- Development of reference mobile wallet
- Development of GUI miner for EtherCore

2020 Q3

- Development of EtherCore sidechain
- Improvement of EtherCore POW mainnet

2020 Q4

- Improvement of EtherCore POA sidechain
- Implementation of mainnet scaling solution

2021 Q1

- Deploy multiple DApps for EtherCore ecosystem
- Light wallet solution for EtherCore
- New Rust lang based EtherCore node

2021 Q2

- Development of Simple Payment Solution for EtherCore
- Improve user experience regarding EtherCore multi-sig wallet

2021 Q3

- Research of EtherCore DeFi solution

2021 Q4

- Implementation of EtherCore DeFi solution
- Integrate 10000+ TPS solution for EtherCore mainnet & sidechain

Glossary

Smart Contract: a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract.

DApp: Decentralized applications, a computer application that runs on a distributed computing system.

PoW: Proof-of-Work, a consensus mechanism that is used to deters denial of service attacks and other service abuses such as spam on a network.

PoA: Proof-of-Authority, a consensus mechanism that is similar to Proof-of-Work, but doesn't requires additional work information from nodes.

Hashing: using a mathematical function to derive a fixed length numeric or alphanumeric string from a piece of data of any length.

Hashrate: a hashrate is the speed at which a computer can solve a hashing problem. The probability of a miner successfully mining a block before anyone else is partially determined by the hashrate of their mining operations.

API: application programming interface.

ASIC: application-specific integrated circuit. A microchip designed specifically for cryptocurrency mining.

GPU: graphics processing unit. a special stream processor used in computer graphics hardware.

BFT: Byzantine Fault tolerance, a property that enables a system to continue operating properly in the event of the failure of (or one or more faults within) some of its components

Terms of Use

This Technical White Paper is for information purposes only. EtherCore does not guarantee the accuracy of or the conclusions reached in this white paper, and this white paper is provided "as is". EtherCore does not make and expressly disclaims all representations and warranties, express, implied, statutory or otherwise, whatsoever, including, but not limited to: (i) warranties of merchantability, fitness for a particular purpose, suitability, usage, title or noninfringement; (ii) that the contents of this white paper are free from error; and (iii) that such contents will not infringe third-party rights. EtherCore and its affiliates shall have no liability for damages of any kind arising out of the use, reference to, or reliance on this white paper or any of the content contained herein, even if advised of the possibility of such damages. In no event will EtherCore or its affiliates be liable to any person or entity for any damages, losses, liabilities, costs or expenses of any kind, whether direct or indirect, consequential, compensatory, incidental, actual, exemplary, punitive or special for the use of, reference to, or reliance on this white paper or any of the content contained herein, including, without limitation, any loss of business, revenues, profits, data, use, goodwill or other intangible losses.

Reference

https://www.tutorialspoint.com/blockchain/index.htm

https://en.wikipedia.org/wiki/Blockchain

https://en.wikipedia.org/wiki/Shard_(database_architecture)

https://www.govtrack.us/congress/bills/116/hres219

https://github.com/ifdefelse/ProgPOW

https://medium.com/@ifdefelse/progpow-progress-da5bb31a651b

https://eips.ethereum.org/EIPS/eip-1057

https://bitcoin.org/bitcoin.pdf

https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

https://www.bitcoincash.org

https://en.wikipedia.org/wiki/Proof_of_work

https://en.wikipedia.org/wiki/Proof_of_authority

https://github.com/poanetwork/wiki/wiki/POA-Network-Whitepaper

https://en.wikipedia.org/wiki/Bitcoin_scalability_problem