Technical Paper

**PLAAK Phaeton Chain**

www.plaak.com

Index

• Introduction

• Basic Terminology

• Peer-to-Peer Network

• Data compression

• Hashing

• Algorithms

• Benchmarks

• History

• Nodes setup & Servers

• Consensus- dPOS

• Side chains & Dapps

• how they work

• Sidechain benefits & how it speeds up the main chain reduces load

• Sidechain validation network (SVN)

• Developer resources

• API Info

• Exchange integration

• Business info & applications

**Abstract**

A general equilibrium monetary model is developed to study the optimal design of a cryptocurrency system based on a blockchain. The model is then calibrated to Bitcoin transaction data to perform a quantitative assessment of the scheme. We formalize the critical elements of a cryptocurrency: the blockchain to keep a history of transactions, the distributed updating of information and consensus through competition for such updating. We show that, unlike cash, a cryptocurrency system does not support an immediate, final settlement. In addition, the current Bitcoin scheme generates a welfare loss of 1.4% of consumption. Such loss can be lowered substantially to 0.08% by adopting the optimal policy which reduces mining and relies on money growth rather than transaction fees to finance mining rewards. The efficiency can potentially be improved further by adopting an alternative consensus protocols such as the delegate-proof-of-stake. A key economic feature of a cryptocurrency system is that mining is a public good, while double spending to defraud the cryptocurrency depends on individual incentives to reverse a particular transaction. As a result, a cryptocurrency works best when the volume of transactions is large relative to the individual transaction size (e.g., as in a retail payment system).

Keywords: Cryptocurrency, Blockchain, Bitcoin, Double Spending, Mining

**Introduction**

Currency transactions between persons or companies are often centralized and controlled by a third party organization. Making a digital payment or currency transfer requires a bank or credit card provider as a middleman to complete the transaction.In addition, a transaction causes a fee from a bank or a credit card company. The same process applies also in several other domains, such as games, music, software etc. The transaction system is typically centralized, and all data and information are controlled and managed by a third party organization, rather than the two principal entities involved in the transaction. Blockchain technology has been developed to solve this issue. The goal of Blockchain technology is to create a decentralized environment where no third party is in control of the transactions and data.

Blockchain is a distributed database solution that maintains a continuously growing list of data records that are confirmed by the nodes participating in it. The data is recorded in a public ledger, including information of every transaction ever completed. Blockchain is a decentralized solution which does not require any third party organization in the middle. The information about every transaction ever completed in Blockchain is shared and available to all nodes.

This attribute makes the system more transparent than centralized transactions involving a third party. In addition, the nodes in Blockchain are all anonymous, which makes it more secure for other nodes to confirm the transactions. Bitcoin was the first application that introduced Blockchain technology. Bitcoin created a decentralized environment for cryptocurrency, where the participants can buy and exchange goods with digital money. However, even though Blockchain seems to be a suitable solution for conducting transactions by using cryptocurrencies,it has still some technical challenges and limitations that need to be studied and addressed. High integrity of transactions and security, as well as privacy of nodes are needed to prevent attacks and attempts to disturb transactions in Blockchain. In addition, confirming transactions in the Blockchain requires a computational power. It is important to identify what topics have been already studied and addressed in Blockchain and what are currently the biggest challenges and limitations that need further studies.

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

While policy makers concern about the opportunities and challenges brought about by these technological advances, there is very little guidance provided by economic theory regarding the appropriate usage of these technologies and the optimal design of these systems. This paper attempts to provide an economic theory to help us understand the fundamental economic trade-offs and address relevant policy issues. Most existing models of cryptocurrencies are built by computer scientists who focus mainly on the feasibility and security of these systems.2 This line of research often ignores the incentives of participants (e.g., the incentives of malicious attackers) and the endogenous nature of key variables (e.g., the real value of cryptocurrencies). More importantly, to study the optimal design of a cryptocurrency system, we need to model from first principles the behaviors of different participants, to derive the equilibrium interactions among these agents and to study the optimal usage of different policy instruments. To this end, this paper develops a general equilibrium monetary model of a cryptocurrency system to study its optimal design. This approach is desirable because the model endogenizes the value of cryptocurrency, and endogenizes the underlying trading activities and mining activities. It also provides a welfare notion for assessing alternative system designs. We will use this model to evaluate the performance of a cryptocurrency system calibrated to Bitcoin transaction statistics. We will study the optimal design of the cryptocurrency system in different settings. Furthermore, we compare the usage of different consensus protocols.

Since the creation of Bitcoin in 2009, numerous private cryptocurrencies have been introduced. Bitcoin is by far the most successful one. It has been getting a lot of media attention, and its total market value has reached 20 billions USD in March 2017. More importantly, a number of central banks started recently to explore the adoption of cryptocurrency and blockchain technologies for retail and large-value payments. For example, the People’s Bank of China aims to develop a nationwide digital currency based on blockchain technology; the Bank of Canada and Monetary Authority of Singapore are studying its usage for interbank payment systems; the Deutsche Bundesbank has developed a preliminary prototype for blockchain-based settlement of financial assets. Many proponents believe that cryptocurrency and blockchain technology will have a significant

influence on the future development of payment and financial systems.

The economic literature on cryptocurrencies is very thin. So far, there are only a few economic models developed to study this new payment technology. These models use different frameworks to address different research questions, and often focus on different aspects of cryptocurrencies.

Chiu and Wong (2015) apply the mechanism design approach to review several e-money technologies including Bitcoin, PayPal and M-Pesa and identify some essential features of e-money that can help implement constrained efficient allocations. Gans and Halaburda (2013) develop a model of platform management to study platform-specific digital currencies such as Facebook Credits.

*Fern´andez-Villaverde* and *Sanches* (2016) model cryptocurrencies as privately issued fiat currencies and analyze whether competition leads to efficiency. *Agarwal and Kimball* (2015) advocate that the adoption of digital currencies can facilitate the implementation of a negative interest rate policy. *Rogoff* (2016) suggests subsidizing the provision of digital money to the unbanked in order to phase out paper currency which facilitates undesirable tax evasion and criminal activities. To the best of our knowledge, our work is the first paper that explicitly models the distinctive technological features of a cryptocurrency system (e.g. blockchain, mining, double-spending problems) in an equilibrium monetary model and investigates its optimal design both qualitatively and quantitatively

**Basic Terminology**

• Blockchain - Is decentralized, public Ledger of Information.

• Consensus - Irrefutable system of agreement between various devices.

• dPOS - Delegated proof of stake uses real-time voting

• Nodehash - Versioning scheme and update mechanism.

• SHA256 - Cryptographic Encryption Algo.

• Hashing - Takes an input and convert into a cryptographic fixed output.

• Seed -

• Digest - Ensuring the security and integrity of the data

• Master node - Genesis Block of Blockchain.

• Delegate - Collecting the transactions across the network into blocks.

• P2p - Utilizes and provides the foundation of the network at the same time.

• SVN - Side chain validation.

Cryptocurrencies: A Brief Review

Bitcoin’s three main technical components: transactions (including scripts), the consensus protocol, and the communication network. Bitcoin is exceedingly complex—our goal is to present the system with sufficient technical depth that the literature on Bitcoin. In particular, a key benefit of our three-component breakdown is that it makes evaluating and systematizing proposed changes insightful by “decoupling” concepts that may be changed independently. As, the idea of exploiting its enabling technology to develop applications beyond currency has been receiving increasing attention. In particular, the public and append-only ledger of transaction (the blockchain) and the decentralized consensus protocol that Bitcoin nodes use to extend it. The archetypal implementation of smart contracts is Ethereum, a platform where they are rendered in a Turing-complete language. The consensus protocol of Ethereum ensures that all and only the valid updates to the contract states are recorded on the blockchain, so ensuring their correct execution.

Besides Bitcoin and Ethereum, a remarkable number of alternative platforms have flourished over the last few years, either implementing crypto-currencies or some forms of smart contracts. For instance, the number of crypto-currencies hosted on coinmarketcap.com has increased from 0 to more than 600 since 2012; the number of github projects related to blockchains and smart contracts has reached, respectively, 2,715 and 445 units. In the meanwhile, ICT companies and some national governments have started dealing with these topics, also with significant investments.

Blockchain technology has also some technical challenges and limitations that have been identified. Seven technical challenges and limitations for the adaptation of Blockchain technology in the future:

• Throughput: The potential throughput of issues in the Bitcoin network is currently maximized to 7tps (transactions per second). Other transaction processing networks are VISA (2,000tps) and Twitter (5,000tps). When the frequency of transactions in Blockchain increases to similar levels, the throughput of the Blockchain network needs to be improved.

• Latency:To create sufficient security for a Bitcoin transaction block, it takes currently roughly 10 minutes to complete one transaction. To achieve efficiency in security, more time has to be spent on a block, because it has to outweigh the cost of double spending attacks. Double-spending is the result of successful spending of money more than once. Bitcoin protects against double spending by verifying each transaction added to the block chain, to ensure that the inputs for the transaction have not been spent previously. This makes latency a big issue in Blockchain currently. Making a block and confirming the transaction should happen in seconds, while maintaining security. To complete a transaction e.g. in VISA takes only a few seconds, which is a huge advantage compared to Blockchain.

• Size and bandwidth:At the moment, the size of a Blockchain in the Bitcoin network is over 50,000MB (February 2016). When the throughput increases to the levels of VISA, Blockchain could grow 214PB in each year. The Bitcoin community assumes that the size of one block is 1MB, and a block is created every ten minutes. Therefore, there is a limitation in the number of transactions that can be handled (on average 500 transaction in one block). If the Blockchain needs to control more transactions, the size and bandwidth issues have to be solved.

• Security:The current Blockchain has a possibility of a 51% attack. In a 51% attack a single entity would have full control of the majority of the network’s mining hash-rate and would be able to manipulate Blockchain. To overcome this issue, more research on security is necessary.

• Wasted resources: Mining Bitcoin wastes huge amounts of energy ($15million/day). The waste in Bitcoin is caused by the Proof-of-Work effort. There are some alternatives in industry fields, such as proof-of-stake, DPOS (Delegate Proof of Stake). With Proof-of-Work, the probability of mining a block depends on the work done by the miner. However, in Proof-of-Stake, the resource that is compared is the amount of Bitcoin a miner holds [12]. For example, someone holding 1% of the Bitcoin can mine 1% of the “Proof-of-Stake blocks”. The issue with wasted resources needs to be solved to have more efficient mining in Blockchain.

• Usability: The Bitcoin API for developing services is difficult to use. There is a need to develop a more developer-friendly API for Blockchain. This could resemble REST APIs.

• Versioning, hard forks, multiple chains: A small chain that consists of a small number of nodes has a higher possibility of a 51% attack. Another issue emerges when chains are split for administrative or versioning purposes

Ethereum, taken as a whole, can be viewed as a transaction-based state machine: It begin with a genesis state and incrementally execute transactions to morph it into some final state. It is this final state which we accept as the canonical “version” of the world of Ethereum. The state can include such information as account balances, reputations, trust arrangements, data pertaining to information of the physical world; in short, anything that can currently be represented by a computer is admissible. Transactions thus represent a valid arc between two states; the ‘valid’ part is important—there exist far more invalid state changes than valid state changes. Invalid state changes might, e.g. be things such as reducing an account balance without an equal and opposite increase elsewhere.

A valid state transition is one which comes about through a transaction. Formally:

σt+1 ≡ Υ(σt, T)

where Υ is the Ethereum state transition function. In Ethereum, Υ, together with σ are considerably more powerful then any existing comparable system; Υ allows components to carry out arbitrary computation, while σ allows components to store arbitrary state between transactions.

Transactions are collated into blocks; blocks are chained together using a cryptographic hash as a means of reference. Blocks function as a journal, recording a series of transactions together with the previous block and an identifier for the final state (though do not store the final state itself—that would be far too big). They also punctuate the transaction series with incentives for nodes to mine. This incentivisation takes places as state transition function, adding value to a nominated account.

Mining is the process of dedicating effort (working) to bolster one series of transactions (a block) over any other potential competitor block. It is achieved thanks to a cryptographically secure proof.

**With Bitcoin:**

Transaction format: A transaction contains an array of inputs and an array of outputs. The entire transaction is hashed using SHA-2564 and this hash eventually serves as its globally unique transaction ID. Transactions are represented using an ad hoc binary format; Each output contains an integer value representing a quantity of the Bitcoin currency. The precision of this value limits the extent to which units of the currency can be subdivided; the smallest unit is called a satoshi. By convention, 108 satoshis is considered the primary unit of currency, called one “bitcoin” and denoted B, BTC or XBT.

Each output also has a short code snippet (in a special scripting language) called the scriptPubKey representing the conditions under which that transaction output can be redeemed, that is, included as an input in a later transaction.

Transaction scripts: Typically, the scriptPubKey specifies the hash of an ECDSA public key and a signature validation routine. This is called a “pay-to-pub-key-hash” transaction and the entire redeeming transaction must be signed using a key with the the specified hash. The vast majority of Bitcoin transactions are pay-to-pub-key-hash and the system is often described with this being the only possibility, although other transaction types are possible. The scripting language is an ad hoc, non-Turing-complete stack language with fewer than 200 commands called opcodes. They include support for cryptographic operations—e.g., hashing data and verifying signatures. Like the transaction format, the scripting language

is only specified by its implementation in bitcoind. Transaction inputs refer to previous transactions by their transaction hash and the index of the output within that transaction’s output array. They must also contain a code snippet which “redeems” that transaction output called the scriptSig. To successfully redeem a previous transaction, the scriptSig and scriptPubKey, must both execute successfully, one after the other, using the same stack. For pay-to-pubkey-hash transactions, the scriptSig is simply a complete public key (with the correct hash) and a signature.

Conservation of value: In addition to the requirements that each transaction input matches a previous transaction output and that the two scripts execute successfully, transactions are only valid if they satisfy the fundamental constraint that the sum of the values of all transaction outputs is

less than or equal to the sum of the values of all inputs.

**With Ethereum:**

The account state comprises the following four fields:

nonce: A scalar value equal to the number of transactions sent from this address or, in the case of accounts with associated code, the number of contract-creations made by this account. For account of address a in state σ, this would be formally denoted σ[a]n.

balance: A scalar value equal to the number of We owned by this address. Formally denoted σ[a]b.

**StorageRoot:** A 256-bit hash of the root node of a Merkle Patricia tree that encodes the storage contents of the account (a mapping between 256-bit integer values), encoded into the trie as a mapping from the 256-bit integer keys to the RLP-encoded 256-bit integer values. The hash is formally denoted σ[a]s.

**CodeHash:** The hash of the EVM code of this account—this is the code that gets executed should this address receive a message call; it is immutable and thus, unlike all other fields, cannot be changed after construction. All such code fragments are contained in the state database under their corresponding hashes for later retrieval. This hash is formally denoted σ[a]c, and thus the code may be denoted as b, given that KEC (b) = σ[a]c.

**Mining:** To understand the consensus mechanism of the Bitcoin system, we first have to discuss the role of a miner. A miner collects pending Bitcoin transactions, verifies their legitimacy, and assembles them into what is known as a “block candidate.” The goal is to earn newly created Bitcoin units through this activity. The miner can succeed in doing this if he or she can convince all other network participants to add his or her block candidate to their copies of the Bitcoin Blockchain.

Bitcoin mining is permissionless. Anyone can become a miner by downloading the respective software and the most recent copy of the Bitcoin Blockchain. In practice, however, there are a few large miners that produce most of the new generally accepted blocks. The reason is that competition has become fierce and only large mining farms with highly specialized hardware and access to cheap electricity can still make a profit from mining.

For a block candidate to be generally accepted, it must fulfill a specific set of predefined criteria. For instance, all included transactions must be legitimate. Another important criterion is the so-called “fingerprint” of the block candidate. A miner obtains this fingerprint by computing the block candidate’s hash value using the hash function SHA256.

For example, we will look at the hash value for the text, “Federal Reserve Bank of Saint Louis.” The fingerprint of this text, which was calculated using the hash function SHA256, is

*72641707ba7c9be334f111ef5238f4a0b355481796fdddfdaac4c5f2320eea68.*

Now notice the small change in the original text to “federal Reserve Bank of Saint Louis.” It will cause an unpredictable change of the fingerprint, which can be seen from the corresponding new hash value:

*423f5dd7246de6faf8b839c41bf46d303014cffa65724ab008431514e36c4dba.*

This characteristic is employed in the mining process as follows. For a block candidate to be accepted by all miners, its fingerprint must possess an extremely rare feature: The hash value must be below a certain threshold value—that is, it must display several zeroes at the beginning of the fingerprint. An example of a fingerprint of a block that was added to the Bitcoin Blockchain in 2010 is given in the following example:

*Block #69785 (July 23rd, 2010, 12:09:36 CET)*

*0000000000 14243 293b78a2833b45d78e97625f6484ddd1accbe0067c2b8f98b57995*

There are M miners performing mining activities to update the public ledger in the night trading sessions n = 0, ..., N¯. In each session, miners perform a costly computational task with a random success rate by investing computing power qn. This task is called the proof of work (PoW). As specified by the Bitcoin protocol, if the computational power of miner i in session n is qn(i), then the probability that a particular miner j will be the first one to solve the proof-of-work problem is given by

Miners are continuously trying to find block candidates that have a hash value satisfying the above mentioned criterion. For this purpose, a block includes a data field (called the nonce) that contains arbitrary data. Miners modify this arbitrary data in order to gain a new fingerprint.

These modifications do not affect the set of included transactions. Just as with our example, every modification results in a new hash value. Most of the time, the hash value lies above the threshold value, and the miner discards the block candidate. If, however, a miner succeeds in creating a block candidate with a hash value below the current threshold value, he or she broadcasts the block candidate as quickly as possible to the network. All the other network participants can then easily verify that the fingerprint satisfies the threshold criterion

by computing it themselves.

**Advanced application features**

**1. Side Chain**

a. Sidechain explorer

With the introduced sidechain standards in Resilience, we can now work on a sidechain explorer which allows users to inspect the blocks, transactions, and account balances of any sidechain. This will provide a consistent, transparent, and easy to access viewpoint of each sidechain to all users.

b. Sidechain security

For security reasons, we shall implement validated hashes of individual sidechain states, at configurable checkpoint intervals. This will ensure there is some basic validation of sidechain integrity, and provide payback to the parent network.

**2. Smart contract support**

Smart contracts are an interesting concept for many different use-cases, and we believe it would be extremely interesting to implement a smart contract in our native App SDK. We want to support Javascript as a language to write Smart Contract. As of now its open for discussion.

**3. Third party technology support**

The beauty of modern software development, is that you don’t have to re-implement every piece of technology from scratch, instead we can use third party libraries and tools. Nowadays, there is already a good number of blockchain technologies available; like Bitcoin, Ethereum, Storj, or Sia. There are also many other interesting technologies available like Torrent, IPFS, and many centralised cloud/computation solutions.

We believe by offering bridges for as many technologies as possible into our App SDK, we will allow developers to build more useful and feature-rich applications.

Once we have started working on this milestone, we will ask our development community what kind of technologies they most want to leverage inside their blockchain applications.

**5. Trustless blockchain applications**

If you have carefully read this blog post until now, you might have realised that you can build great blockchain applications of nearly any kind. However, until this point blockchain application developers are still able to exercise a large amount of control over their sidechains, possibly to the detriment of decentralization.

This kind of control is the preferred solution for startups who want to keep control of development, and build a business model around their blockchain applications, while also enjoying the other benefits of blockchain technology. However, some developers want to develop truly trust-less blockchain applications. Applications which can’t be shut down by any single entity, such as the developer, and are secured by anyone who wants to participate.

**6. Proof of Stake**

We think that a Delegated Proof of Stake consensus algorithm is the best method to bootstrap a sidechain, as it is extremely efficient, and secured by a number of incentivized (=paid) actors, and maintains a degree of control over its development cycle in it’s early stages. Which in our view, provides the best solution for startups who want to build blockchain applications.

However, in order to achieve that, a delegate marketplace must be available, and the blockchain application developer, or its users are required to pay the delegates for securing the sidechain.

We also believe that a trust-less voting mechanism, as our ecosystem has on the mainchain, can’t scale effectively to hundreds of sidechains, as the lack of incentive to vote, and voter apathy would eventually hinder it’s long-term security.

Therefore, a Proof of Stake consensus algorithm, where everyone can simply plug into the sidechain, and start securing it, is the best method for truly trust-less blockchain applications, where even the developer themselves doesn’t have authority over consensus.

This would mean developers can not rent node operators on the delegate marketplace for their trust-less blockchain application. It would be the other way around, where we release a Sidechain Marketplace in which these trust-less blockchain applications can be promoted. Node operators are then able to browse and find sidechains they can secure. The incentive for doing so, would come from the potential to earn transaction fees, and optionally block rewards in the form of custom tokens.

**7. Security, optimisation, scalability, and possible consensus changes**

We will scale up and secure the whole network to such a level that it is enterprise and finance ready. Until now the App SDK was mainly targeting consumer applications, after this phase and the milestones are achieved, this will obviously change.

**8. Decentralised Voting Mechanism**

Everyone is able to add surveys (e.g. to get opinions on their ideas) and every app share holder can express their opinions, or vote for the different options. This way the whole community can practice decentralized decision making without any centralised entity dictating the outcome.

As another step, everyone can add proposals (e.g. for a blockchain application or core feature) and request PLKX from the fund. If a proposal reaches consensus (enough yes votes) the curators get notified, and can release the necessary funds. This way the community can continue the development of ecosystem, without any centralised team, and without any centralised decision making.

**Consensus Mechanism:**

The consensus among miners is that every miner who receives a block candidate with a valid fingerprint adds it to his or her own copy of the Bitcoin Blockchain. From a game theoretical perspective, a strategy profile where all miners add valid blocks to their own copies of the Bitcoin Blockchain is a Nash equilibrium.

Mining is expensive, as the computations use large amounts of electricity and are increasingly dependent on highly specialized hardware. Moreover, valid block candidates can be found only through a trial-and-error procedure. The consensus mechanism is therefore called “proof of work.” If a miner finds a valid fingerprint for a block candidate, then this is proof that he or she has, on average, performed a large number of costly computations. Adding false information (e.g., illegitimate transactions) to a block candidate would render the block candidate invalid and essentially waste all the computations. Finding a valid fingerprint is therefore proof that the miner helped to maintain the Bitcoin system.

With Phaeton, we uses Delegated Proof of Stake (DPoS) as its consensus protocol. Delegates generate all of the blocks within the system and are elected by the stakeholders, in this case all entities holding LSK tokens. The number of delegates is fixed at 101. Each stakeholder can vote for up to 101 delegates, and the weight of the vote depends on the amount of LSK the stakeholder possesses. Any stakeholder can vote for a delegate using a vote transaction.

Consensus is a key aspect of any blockchain system. It serves a vital purpose in a system where there are countless nodes and all nodes need to agree on the integrity of the data that is being recorded on a blockchain. As a term, ‘consensus’ means that the nodes on the network agree on the same state of a blockchain, in a sense making it a self-auditing ecosystem. This is an absolutely crucial aspect of the technology, carrying out two key functions. Firstly, consensus protocols allow a blockchain to be updated, while ensuring that every block in the chain is true as well as keeping participants incentivized. Secondly, it prevents any single entity from controlling or derailing the whole blockchain system. The aim of consensus rules is to guarantee a single chain is used and followed.

**P2P Network**

Peer-to-peer architecture (P2P architecture) is a commonly used computer networking architecture in which each workstation, or node, has the same capabilities and responsibilities. It is often compared and contrasted to the classic client/server architecture, in which some computers are dedicated to serving others.

A peer-to-peer network is designed around the notion of equal *peer* nodes simultaneously functioning as both "clients" and "servers" to the other nodes on the network. This model of network arrangement differs from the client–server model where communication is usually to and from a central server. A typical example of a file transfer that uses the client-server model is the File Transfer Protocol (FTP) service in which the client and server programs are distinct: the clients initiate the transfer, and the servers satisfy these requests.

Peer-to-Peer communication serves a vital function within the Phaeton network. The peering mechanisms provide the required architecture to facilitate network consensus, block propagation and transaction propagation.

**- Architecture**

The peers in the Phaeton network use JSON objects with compressed blocks and transactions. The Phaeton logic running on every node in the Phaeton network uses remote procedure calls (RPCs) and events to communicate the transaction and block JSON objects to the other peers. The RPCs and events are also transmitted as JSON objects with additional fields telling the Phaeton application which method to use in order to process the transmitted object. In order to effectively transmit these JSON objects to the other peers, websockets are used via the SocketCluster Framework.

**- System Headers**

Every time a Phaeton node communicates with a peer of the Phaeton network, a system header is added to the message. The system headers are used to identify full nodes and provide basic information about the software running on the system.

The following JSON object is generated from system data for this purpose:

{

"os":"darwin16.3.0",

"version":"0.6.0a",

"port":7000,

"height":1574654,

"nethash":"da3ed6a45429278bac2666961289ca17ad86595d33b31037615d4b8e8f158bba",

"broadhash":"c7e0902a7016205d456a427edda2b09f4b875f98ef40a224018a0274347146ac",

"minVersion":">=0.5.0"

}

**- Block Propagation**

Block propagation serves a vital function on the Phaeton network. Without block propagation, the system would grind to a halt and the blockchain would cease to function. Blocks are made in a decentralized fashion and must be sent to all nodes on the network in order to establish consensus. When a block is generated, it is broadcast to 25 randomly selected peers. These forward the validated block to 25 randomly selected peers and so on. In order to prevent over broadcasting of data, every block is given a relay limit of 3 and blocks that have already been received are not broadcast again.

**- Transaction Propagation**

Transactions must move from one node to all other nodes in order to be included in blocks. The broadcast queue for transactions works by drawing up to 25 transactions from the transactions pool and performing a validation process on those transactions. These transactions are then broadcast to other nodes in a bundled JSON object. This can be represented as an array of objects, depending on the transaction type. The bundle is then broadcast to the network at regular intervals, currently specified as every 5 seconds. The time delay allows the bundle to accumulate additional transactions from the network (up to 25). In addition to broadcasting the object, the bundle is given a relay limit to prevent spamming the network. In the current implementation the relay limit is set as 3, which means that every bundle will be broadcast for at most 3 hops by the peers on the network.

**- Transaction Pool**

The transaction pool provides the Phaeton network a robust solution for preserving unconfirmed transactions that have overflowed into the next block. As described in blocks, each block can only include 25 transactions and the transaction pool allows up to 1.000 multi signature transactions and other 1.000 for the remaining transaction types to remain queued for the next block(s). The transaction pool could be thought of as a memory pool, keeping transactions ready until they are signed into a block. The second usage of the transaction pool is to provide a mechanism for propagating transactions. When a node prepares a transaction bundle, it draws up to 25 transactions from the pool and broadcast them to the network. In order to keep the transaction pool tidy, all transactions are given a time to live. This time to live is defined as 10800 seconds, or 1080 blocks. The final use for the transaction pool is to house transactions with pending signatures. Like unconfirmed transactions, these transactions will expire out of the pool based on the lifetime specified when the transaction is first received.

**Data Compression:**

Data compression is known for reducing storage and communication costs. It involves transforming data of a given format, called source message, to data of a smaller sized format, called codeword. Data encryption is known for protecting information from eavesdropping. It transforms data of a given format, called plaintext, to another format, called cipher text, using an encryption key. The major problem existing with the current compression and encryption methods is the large amount of processing time required by the computer to perform the tasks. To lessen the problem, I combine the two processes into one.

Phaeton uses elliptic curve cryptography and cryptographic hashing in order to secure all aspects of the system. The system uses EdDSA as it provides a robust and fast mechanism for hashing and providing security.

**- Key Pair**

A key pair consists of a private key and a public key. A private key is a string of numbers and letters only known to the owner of the key. The public key is derived from the private key and can be used to validate that the private key belongs to the owner without providing access to their private key. Elliptic curve cryptography is used to generate cryptographically secure key pairs.

The process used to generate the key pair operates in the following manner:

When a user creates an account, a BIP39 mnemonics (the passphrase) is generated for the user. This passphrase is hashed using the BLAKE2B hash function into a 256-bit string. This hash is subsequently used as a seed in Ed25519 to generate the private key ks and derive its public key. With this private key, the user is able to sign transactions into a transaction object and broadcast that object to the network. The public key is included as part of the transaction and the nodes that receive the transaction are able to verify the validity of the signature using kp. This provides effective security for both the user and the network since ks is known only to the user and kp can validate that the signature is valid.

**- Address**

An address or the wallet ID is derived from the public key. The public key is hashed using BLAKE2B, at which point the first 8 bytes of the hash are reversed. The account ID is the numerical representation of those 8 bytes, with the ’L’ character appended at the end. The following figure is the representation of an address and its associated account details.

*{*

*"address": "16009998050678037905L",*

*"unconfirmedBalance": "0",*

*"balance": "0",*

*"publicKey": "73ec4adbd8f99f0d46794aeda3c3d86b245bd9d27be2b282cdd38ad21988556b",*

*"unconfirmedSignature": 0,*

*"secondSignature": 0,*

*"secondPublicKey": null,*

*"multisignatures": [],*

*"u\_multisignatures": []*

*}*

**- Second passphrase**

Phaeton also offers the option of an additional layer of security. Using a specific type of transaction, the user can register a second passphrase that is associated with the account. This relationship requires all subsequent transactions to be additionally signed using the second passphrase in order to be considered valid. The process of generating the second key pair is the same as the one for the initial key pair.

- **Multisignature**

Phaeton also supports multisignature accounts as another security system for users requiring even greater security. A multisignature account is an account that requires multiple keys to authorize a transaction. Any user can enable multisignature on their account by issuing a special transaction specifying a group of n keys and the minimum number m of signatures required to authorize a transaction. Once this is done, it is mandatory that any transactions originating from that account must be signed by at least m out of the n keys for the transaction to be processed.

**Hashing:**

* **Elliptic Curve Cryptography**

Phaeton uses ECC (Elliptic Curve Cryptography) to sign digital assets to ensure the security of every transaction. The public key can be calculated from a known private key. ECC is widely regarded as the most powerful asymmetric algorithm given the key length, which has been fully exercised in the bitcoin network.

Elliptic curve cryptography (ECC) is an approach to public key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non ECC cryptography (based on plain Galois fields) to provide equivalent security. Elliptic curves are applicable for key agreements, digital signatures, pseudo random generators and other tasks. Indirectly, they can be used for encryption by combining the key agreement with a symmetric encryption scheme.

**- Algorithms:** BLAKE2 is a cryptographic hash function faster than MD5, SHA-1, SHA-2, and SHA-3, yet is at least as secure as the latest standard SHA-3. BLAKE2 has been adopted by many projects due to its high speed, security, and simplicity.

BLAKE2 is specified in RFC 7693, and our code and test vectors are available on GitHub, licensed under CC0 (public domain-like). BLAKE2 is also described in the 2015 book The Hash Function BLAKE.

BLAKE2 comes in two flavors:

BLAKE2b (or just BLAKE2) is optimized for 64-bit platforms—including NEON-enabled ARMs—and produces digests of any size between 1 and 64 bytes

BLAKE2s is optimized for 8- to 32-bit platforms and produces digests of any size between 1 and 32 bytes

BLAKE2 includes the 4-way parallel BLAKE2bp and 8-way parallel BLAKE2sp designed for increased performance on multicore or SIMD CPUs. BLAKE2 offers these algorithms tuned to your specific requirements, such as keyed hashing (that is, MAC or PRF), hashing with a salt, updatable or incremental tree-hashing, or any combination thereof. These versions are specified in the BLAKE2 document.

BLAKE2 also includes the BLAKE2x variants, which can produce digests of arbitrary length. BLAKE2x is specified in a separate document.

BLAKE2 shines on 64-bit CPUs: on an Intel Core i5-6600 (Skylake microarchitecture, 3310MHz), BLAKE2b can process 1 gibibyte per second, or a speed rate of 3.08 cycles per byte.

The plot below shows how BLAKE2 outperforms MD5, SHA-1, SHA-2, and SHA-3 on a Skylake Intel CPU (speeds are for hashing using a single core; using multiple cores, BLAKE2 can be even faster):

**- History**

Since BLAKE2 is very similar to BLAKE, we first describe the changes introduced with BLAKE2. We refer to https://131002.net/blake for a complete specification of BLAKE.

2.1 Fewer rounds

BLAKE2b does 12 rounds, and BLAKE2s does 10 rounds, against 16 and 14 respectively for BLAKE. Based on the security analysis performed so far, and on reasonable assumptionson future progress, it is unlikely that 16 and 14 rounds are meaningfully more secure than 12 and 10 rounds. Recall that the initial BLAKE submission had 14 and 10 rounds, respectively, and that the later increase was motivated by the high speed of BLAKE.

This change gives a direct speed-up of about 25% and 29%, respectively, on long data. Speed on short data also significantly improves.

2.2 Rotations optimized for speed

The G function of BLAKE-512 performs four 64-bit word rotations of respectively 32, 25, 16, and 11 bits. BLAKE2b replaces 25 with 24, and 11 with 63:

• Using a 24-bit rotation allows SSSE3-capable CPUs to perform two rotations in parallel with a single SIMD instruction (namely, pshufb), whereas two shifts plus a logical OR are required for a rotation of 25 bits. This reduces the arithmetic cost of the G function, in recent Intel CPUs, from 18 single cycle instructions to 16 instructions, a 12% decrease.

• A 63-bit rotation can be implemented as an addition (doubling) and a shift followed by a logical OR. This provides a slight speed-up on platforms where addition and shift can be realized in parallel but not two shifts (i.e., some recent Intel CPUs). Additionally, since a rotation right by 63 is equal to a rotation left by 1, this may be slightly faster in some architectures where 1 is treated as a special case. No platform suffers from these changes. For an in-depth analysis of optimized implementations of rotations, we refer to a previous work by two co-designers of BLAKE2. Past experiments by the BLAKE designers as well as third parties suggest that known differential attacks are unlikely to get significantly better, nor worse (cf. §4).

2.3 Minimal padding and finalization flags

BLAKE2 pads the last data block if and only if necessary, with null bytes. If the data length is a multiple of the block length, no padding byte is added.

BLAKE2 introduces finalization flags f0 and f1, as auxiliary inputs to the compression function:

• The security functionality of the padding is transferred to a finalization flag f0, a word set to ff...ff if the block processed is the last, and to 00...00 otherwise. The flag f0 is 64-bit for BLAKE2b, and 32-bit for BLAKE2s.

• A second finalization flag f1 is used to signal the last node of a layer in tree-hashing modes. When processing the last block—that is, when f0 is ff...ff—the flag f1 issues also set to ff...ff if the node considered is the last, and to 00...00 otherwise. The finalization flags are processed by the compression function as described in §2.4.BLAKE2s thus supports hashing of data of at most 64 − 1 bytes, that is, almost 16 exbibytes (the amount of memory addressable by 64-bit processors). The upper bound for BLAKE2b is even more ridiculous, with up to 128 − 1 bytes supported.

2.4 Fewer constants

Whereas BLAKE used 8 word constants as IV plus 16 word constants for use in the compression function, BLAKE2 uses a total of 8 word constants, instead of 24. This saves 128 ROM bytes and 128 RAM bytes in BLAKE2b implementations, and 64 ROM bytes and 64 RAM bytes in BLAKE2s implementations. The compression function initialization phase is modified to:

**Concense Delegated Proof of Stake:**

Delegated Proof of Stake (otherwise known as DPoS) is a consensus algorithm maintaining irrefutable agreement on the truth across the network, validating transactions and acting as a form of digital democracy. It is the protocol of choice at our platform and with very good reason.

The DPOS algorithm is divided into two parts: electing a group of block producers and scheduling production. The election process makes sure that stakeholders are ultimately in control because stakeholders lose the most when the network does not operate smoothly. How people are elected has little impact on how consensus is achieved on a minute by minute basis. Therefore, this document will focus on how consensus is reached after the block producers have been chosen.

**Process of Delegated Proof-of-Stake**

The process of Delegated Proof-of-Stake varies from more traditional consensus mechanisms. In DPoS, stakeholders elect what are known as witnesses. Witnesses are responsible and rewarded for generating and adding blocks to the blockchain. Each stakeholder is only permitted one vote per witness, with witnesses with the most votes being elected. Stakeholders can vote for as many witnesses as they wish, so long as at least 50% of the stakeholders believe sufficient decentralization has been achieved through the number of elected witnesses. The voting for witnesses is a continuous process, therefore, witnesses have an incentive to carry out their function to the highest standard or they risk losing their position. A reputation scoring system exists in order to assist stakeholders in better assessing the quality of witnesses.

Depending on the cryptocurrency implementing the DPoS consensus mechanism, a chosen group of witnesses are typically replaced at a fixed time, e.g. once a day or once a week, with each witness being given a turn to produce a block. Failure to produce a block at the allocated time will typically result in a witness being skipped, as well as negatively affecting their reputation score.

Delegated proof of stake uses real-time voting combined with a social system of reputation to achieve consensus. It can be seen to be the least centralized consensus protocol compared to all others as it is the most inclusive. Every token holder can exercise a degree of influence about what happens on the network.

Active delegates are voted into their roles by token holders. The voting power that the token holder has, otherwise known as voting weight, is determined by how many of the base token the account is holding. It is important that the delegates are chosen with the best interest of the network at heart as they keep the network running smoothly and safely. In some DPoS versions, a delegate needs to show commitment by depositing his funds into a time-locked security account (which is confiscated in case of malicious behavior). This version of DPoS is often referred to as deposit-based proof of stake.

The roles of delegates revolve around:

Ensuring their node is always up and running.

Collecting the transactions across the network into blocks.

Signing and broadcasting those blocks, validating the transactions.

If there are issues in regard to consensus, DPoS allows these to be resolved in a fair and democratic way.

Delegates do not have the power to change any transaction details. However, as they are validators they could theoretically exclude certain transactions in a block. Nevertheless, this has very little effect as the next created block will include these transactions, giving the next delegate the fees associated with validating them. As such, the transactions will only be slightly delayed. Furthermore, this would inevitably lead to the dishonest delegate getting voted out by the rest of the network. In essence, a DPoS network is self-governed and policed by all of its participants ensuring the best interests of the network remain the priority.

**Advantages of Delegated Proof-of-Stake:**

Delegated Proof-of-Stake offers advantages over the more well-known consensus algorithm, Proof-of-Work (PoW). These advantages include:

Saving on energy costs

Promotes decentralization

Savings on energy costs: In contrast to PoW, which requires large amounts of energy in-order to decide who gets to add the next block to the blockchain, with DPoS, witnesses are given a specific time schedule to do so. Therefore, the intense competition for the addition of the next block becomes impractical, which in turn reduces the energy costs for adding a single block when compared to PoW. Specialized computers known as, ASICs, are no longer necessary in-order to solve complex mathematical problems needed for PoW.

**Promotes decentralization:** In order to successfully mine a cryptocurrency that uses the PoW consensus mechanism, realistically, ASICs, are required in order to be competitive and increase ones chances of adding the next block. This promotes centralization of mining because only those that can afford these specialized computers will stand the best chance of finding a valid block, and thus, reap a larger proportion of the block rewards. This is in contrast with a DPoS consensus mechanism, which allows stakeholders to choose who gets to validate transactions, therefore promoting greater decentralization. An unlimited number of individual validators can be elected by stakeholders so far as they believe sufficient decentralization has been achieved.

**AWS Infrastructure:**

Genesis Instance is create in AWS EC2 Instance. Few Instruction need to follow as mentioned below:

- Switch to Root user as “su”.

- Update Ubuntu version will prepare instance to work well with node setup.

- Install Phaeton From Git clone.

- Install Nodejs 6.14.1.

- Install Postgres Version 9.6.

- Install Phaeton Node by issuing command in root folder “npm install”.

- Open Config/mainnet/config.js file and put required settings.

- After success fully installation issue command “pm2 start —name plaak app.js -n main net”.

- This will start generating main node and sync genesis block parameters to help setup current node.

- We are good to go with this.

Code backup and code deployment is automated and versioning of is managed in secure infrastructure.

**Nodes setup & Servers:**

Over all system is designed with help of Nodejs, React, AngularJS. All three of these platform/framework are known for there usability, fast access and secure communication with backend. Major of these technology are secure with regards to attaches and we make sure that sour system will stand still with any loose to our customer.

This document details the prerequisites to install Phaeton Core 1.0.0 from a Source installation using tagged releases on Github.

To complete the installation there are prerequisites that need to be fulfilled. If you have already performed these, please proceed to the Installation page.

**Steps:**

Open necessary ports.

Create a new user.

Install Tool chain components.

Git Installation.

Node.js Installation.

Recommended: Install NVM.

Recommended: Install PM2.

Postgres Installation.

Redis Installation: Port 6380.

Once all these pre-requisites are in please install Phaeton core from Bitbucket and run "npm install" to install all required node\_modules for this platform.

After success-full installation of all requirements from package.json, To test that Phaeton Core is built and configured correctly, issue the following command:

> node app.js

If the process is running correctly, no errors are thrown in the logs.

By default, errors will be logged in logs/plaak.log only. You can change the logging level in config.json. Once the process is verified as running correctly, CTRL+C and start the process with pm2. This will fork the process into the background and automatically recover the process if it fails.

> pm2 start --name plaak app.js -- -n mainnet

We are good to go with our Node setup.