Braincell Whitepaper

Abstract

The intent of Braincell/zkBrain is to provide an alternative network which provides early implementation of particularly useful second layer system blockchain technology; trading off quick and dirty first layer (L1) ideas for scaling that could create a more centralized system for a more stable, secure, and accessible “state”. Braincell accomplishes this through the use of clever protocols which rest on top of the blockchain, keeping the blockchain architecture itself as is while implementing ZK-S[NT]ARKs technology via rollup-style chain which still makes use of the blockchain and therefore inherits many of the security benefits and properties of the chains which Braincell is built on top of through the advancement of the application layer. Braincell has an implementation on the Ethereum network where in certain security features are secured not just by mainnet Ethereum, but also security features implemented by the Braincell foundation and secured through a PoS consensus mechanism by the Braincell token. The foundation also intends to provide cross-chain bridges to aid interoperability between networks, as well as Braincell wallets, UI friendly browser and mobile extensions and payment interfaces and protocols that implement zk-s[nt]ark technology to create both a privacy preserving means to transact and for nearly instant transaction verification through the verification of transactions at randomly selected checkpoints in the roll-up and smart contracts to safely hold deposited funds, components that allow for network composability, and other forward compatible solutions to scale.

Braincell (Neura) is the ERC-20 token which the Braincell network; acting as the asset to be used under the proof of stake consensus mechanism in the Ethereum based rollup to start.

Necessity of Scale

Several base layer blockchains have been criticized as of recent not only for their extended block confirmation times, amongst other centrally controlled parameters, but also for their increasingly exorbitant transaction fees, especially during times of above average network load. This issue is no short lived one at that; it is an ongoing problem faced by several base layer blockchains as the complexities of these chains increases to accommodate the DApps built within their ecosystems. As a result, the argument of scaling blockchains has very quickly become a major – if not the most discussed issue – in the blockchain space.

The major debate in the space is not the necessity of scale – there seems to be well documented consensus on that – it is the method of scale taken up to do so.

Base layer blockchains are mostly limited to scaling via block size increases or through the use of improvement mechanisms that involve the hastening of message and transaction processing by making the underlying blockchain itself capable of processing increased amounts of transactions

at a faster rate despite the limiting capacity of participating nodes – as no blockchain can process more transactions than a single node – through the modification of the blockchain protocol *or* to how clients themselves work on the blockchain (eg sharding). One could even give up altogether on scaling blockchains individually and make it the responsibility of applications to find and build on top of the appropriate blockchains to best serve them.

By far the most common and simple solution to scale is to simply do so without thought or regard for the any of the community members and contributors or the community as a whole through the adjustment of block size parameters; specifically, through the proposal and implementation of increased block size limits. Which we consider scaling "without thought", or without regard for the community or the blockchains ecosystem itself. While setting a block size limit is considered a central planned parameter similar to other parameters such as the average target time between proposed blocks, its effects on scale are felt much more directly.

The fundamental narrative in scaling through first layer systems seems to lie in weighing the tradeoffs between scalability, security, and decentralization.

The arguments for block size parameter constraint and expansion are lined throughout every past discussion in the community relating to scaling and even still there exist ongoing block size disagreements.
As far as the block size debate is concerned there must always be some tradeoff between the ease of reading the blockchain as a node and writing to the blockchain.
(by "reading" the blockchain we refer to the action of running a node on the network that can verify additions to the ledger, messages, transactions, and the order which transactions were processed, and by "writing" we simply refer to sending transactions and signing them through one's secret or private key.) In order for any blockchain or even second layer system to truly be ascribed as decentralized, it is of the utmost that nodes find the task of reading and writing to the network relatively easy in respect to existing relatively attainable consumer hardware.

Take for instance the state of the bitcoin network prior to Satoshi's protocol changes in the bitcoin network code in 2010 to make blocks larger than 1 million bytes (1MB) invalid. While there was no intentional block size to begin with, the initial block size limit of 32MBs was merely attributed to the network as that was the size limit of peer-to-peer messages. Should this block size limit of 32MBs have been retained the ability for nodes in the network to read the blockchain would have become a significantly more difficult as with such large block size parameters reading the network becomes very resource intensive and financially draining particularly for nodes reliant on consumer hardware which simultaneously and inevitably forces them to drop out as validators with time. In such networks where block size parameters are large the network shifts towards reliance on a significantly smaller subset of nodes using supercomputers to run the blockchain which leads directly to increased exposure to centralization risks.
In such a case, the vast majority of the network – which, due to a lack of resources – becomes incapable of reading the blockchain because of the large block size and the increased difficulty in participation for nodes that comes as a result of increasing the parameter. Should a situation where the vast majority of the participants on a blockchain are incapable of reading the chain due

to a lack of qualifying infrastructure the participants will simply rely on the validations performed by nodes that can read the chain and should there exist a chain where these nodes were to be malicious, they could change the rules of the network without the permission of other nodes with ease as they are the only ones with the ability to read the chain as it stands.

Contrarily, should the block size parameter of a chain have a relatively small limit – a few thousand bytes for example – then the chain becomes increasing resource intensive and costly to write to as miners simply add transactions to the Merkle tree that simply pay them the most regardless of the fact that all full nodes will have to process these transactions. Should this take place, then users of the chain will inevitably rotate to second layer systems where centralization is embraced to combat their exorbitant cost of writing to these chains with very small block size limits. One could adopt the same method as the Bitcoin users in an attempt to satisfy both opposing parties, segwit, but even that leaves much to be desired. This is not to say that first layer solutions to scale do not have their place, but simply that scaling through the mindless increasing of parameters like block size are neither viable nor forward compatible solutions especially with the coercive use of soft forks to make these protocol changes. As such Braincell will implement small blocks to allow for faster block propagation times and nearly instant transactions.

While there are still many legitimate grievances in the community in regards to block size parameters especially with their implementation through soft forks in existing protocols, as soft forks are protocol changes that affect every validator on the network regardless of their view towards the occurrence of such a fork as an opposing constituency, scalability without increasing block size parameter can take place and the technology at the forefront seems to be sharding.

In a blockchain where sharding is not implemented (a vast majority of base layer chains) every node on the network is required to store the entire state of the blockchain as full nodes, and while there exist other node types aside from full nodes such as the light node (implemented with SPV – simplified payment verification) these nodes do not verify the current state as a full node would (i.e., they do not fully validate the block but rather *trust* that the proposed block is objective truth until proven otherwise). And while archival nodes are available of protocols like Ethereum they are simply intended to index all data referring to the historical state of every single block. In essence these archival nodes are only a necessity to those who find it absolutely necessary (e.g., block explorers) to have an archive of previous block data. But, if necessary, an archival node can be constructed from a single full node that is not archival. The introduction of light clients which do not fully validate blocks (as they not to verify the current state) introduces a vulnerability creating the possibility that these clients could be fed false or invalid block information from malicious full-nodes on the network; this emerges as a result of these clients not processing every transaction in a block (ie not fully validating the block) and rather *trusting* that the proposed block is correct until proven otherwise.

Sharding in this sense refers to a change in the blockchain architecture where individual full nodes no longer need to process every single transaction and verify the entirety of the Merkle tree, but instead are only required to process a very small portion of transactions and store a

relatively small portion of data through grouping via methods such as random sampling to create these groups.

Should a blockchain exist where scale is necessary for the chain to be able to process 6,900 tx/s, but the limiting capacity of participating nodes will only allow for 100tx/s then should these nodes be grouped randomly in such a way that potentially malicious nodes are also randomly divided and it is assumed that the majority of the network is mostly honest or rational in many instances then each group/shard could simply be assigned a block to validate; providing a majority signed message validating that the block itself is correct and proceeding to broadcast that signed message to other participating nodes rather than every node processing every transaction within every block and storing large amounts of data (because if 6,900tx/s is the current target for a network, this could easily necessitate the use of supercomputer nodes where large block sizes are implemented and also the storing of several hundred gigabytes of data relating to the state of the network). All other participating nodes would only need to verify the group signed signature rather than the blocks themselves directly.

While nodes may not *directly* be convinced of the validity of the block itself one can be assured that in a randomly selected committee of validators who have been assigned with the task of creating shard blocks (which could be some arbitrary number like 100 nodes in a committee depending on the number of validators/stakers active on the network) on a chain where validators are rational and have staked their resources and at least 67% have reached consensus on the validity of the proposed block then the proposed block is increasingly likely to be valid and can be taken as such. By pairing these shards with the appropriate logic to check and manage these shards and by balancing the number of existing shards with the cost of managing each one, significant scale is also furthered.

In the same way that there exist full, archival and light nodes in the Ethereum ecosystem a system where blockchain architecture is changed in a way to allow for sharding to exist there exist other types of nodes. But as this is beyond our scope of discussion. This first layer system of scale, sharding brings to the forefront the usefulness of probability in scaling blockchains. In the same way that one cannot be *fully* convinced without any doubt that computation is accurate, it is possible to be convinced of the probability of some computation being valid and use such to create a scalable blockchain. This is not exclusive to first layer system; second layer systems also have such means for scale.

Paradigms for Scale

While both the Bitcoin and Ethereum experiments can be considered relatively successful as the pioneering platforms in the decentralized platform space, having gathered massive attention and relatively rapid adoption with Bitcoin being the digital currency of choice and Ethereum being the most widely used smart contract platform with its use of contract accounts and autonomous agents; even so, with various plausibly proposed and even widely utilitarian applications for blockchain technologies underworks or already in use, a set roadblock to the mass adoption of cryptographic assets, DApps, financial contracts, and the likes still remains in place. This

roadblock being the propagation time for blocks, the necessary amount of time needed to generate a valid proof for the validity of a block and the transactions contained within that block, along with the increased fees imposed during times of peak network load to offset confirmation times (and often times this is as a result of increased volume in usage or attempted adoption of new and innovative blockchain applications on a particular mainchains) is still relevant. But as stated earlier the current proposed tradeoff to these problems usually revolves around some form of implemented centralization where certain nodes could be delegated as validators.

Some proposed solutions intended to solve for the issue of scale on Mainnets involve building an entirely new protocol for each new application, a fundamentally flawed immediate decision when creating DApps or blockchain applications of similar nature which neglects both the existing community aspects, and the technological possibility that the encountered problems may still be solved using existing protocols, or, in the very least, solved using improved implementations of existing protocols. Another widely suggested solution is the addition of a group of verified centralized validators for a blockchain, but this also suggest that moving away from the decentralized nature of blockchains in favor of lower transaction fees and faster block propagation time as a reasonable solution; ignoring the base idea of a blockchain's need to eliminate trust.

Prior to this point we have addressed block propagation times in regard to the usage of larger blocks and smaller block and the tradeoffs that come with each, and while various grievances of each side have not been addressed in a manner that either side feels that its argument has been heard; but even still with the adoption of small block sizes (which allow for the shortest block confirmation times of the two) and the *relatively short* block propagation times that come with the adoption of smaller block sizes, transactions are still required to wait for extended periods of time for block confirmations even while experiencing both relatively high propagation rate and relatively high stale rate relative to the networks. The major problem here being that newly propagated blocks containing transactions cannot be immediately confirmed – and even take longer amounts of time on a proof of work-based chain – as the risk of chain re-organization still exist with increased likelihood. Should a transaction (T1) be included in a propagated block in the chain and several blocks then be built on top of the block which included that transaction (T1) then a node can be convinced of the validity of the block which included T1 as the block ages and even more blocks are built on top of it. But with just this method of block propagation there is no way to have nearly instant transaction confirmations which are necessary to met the demand and scale the is required of blockchains for more practical application.

It is often due to several Layer 1 (L1s) blockchains use of consensus mechanisms which revolve around POW that create extended block propagation times and in turn extended confirmation times.

The original proof of work implementation proposal of PoW by Dwork and Anor can be seen ***here***, http://www.hashcash.org/papers/pvp.pdf , which was originally an attempt at combating spam and junk mail in a time when email served as the dominant form of communication online. The overview of the experiment being that to send an email a proof of work would need to be attached by the sender, and while this would make sending mails relatively cheap for the sender

to a set of identifiable and intended individuals; the requirement for computational proof of work would come at a cost of computational resources making so uneconomically practically for email spammers who have no specific, identifiable person or set of persons which they send emails to that their operation would become financially infeasible. The idea fell flat mainly due to the requirement of economic cost to attach a proof of work to emails and the additional amount of time needed and computational resources needed to do so as emails are still a free service today and the existence of economic cost and time element necessary in PoW for emails deters many from using such a service that requires it.

By introducing a Proof of Stake (PoS) mechanism into the consensus layer it becomes possible to offset several of the limitations that come with the use of PoW and with enough distributed stakers the blockchain can also remain decentralized rather than having to adjust for centralized control. Also, the introduction of a slashing and bounty mechanism for malicious actors and those who notice and provide valid proof of the malicious behavior becomes a consequence for both the malicious user and identifier of the malicious behavior respectively. This security mechanism involving bounty for spotting malicious behavior becomes an ongoing open contract ensuring security as nodes are constantly bounty hunting in an attempt to receive a (portion of) another's stake as a reward. This creates a security incentive for validators and ensures continued decentralization in public blockchains.

Genesis: ZK-S[NT]ARKS

The elementary idea and use cases behind zk-s[nt]arks is quite appealing and apparently useful once questions and biases in regards to whether or not the technology even exist or can successfully be deployed have been alleviated.

Their appeal lies in the technology that allows for the verification of the validity of some complex computation without having to actually fully validate the computation itself by redoing said computational work or even knowing exactly what computation took place. Where zk-s[nt]arks are implemented, one must simply verify the validity of the provided proof rather than the computation itself and check to see if the output state post STF (state transition function) is valid along with other such checks. zk-s[nt]arks can be used to combat the problem of scaling, through this means. In using snarks and snark logic on first layer systems one must recognize that they cannot be a replacement for sharding and the use of shard blocks. But they can be used to store the most current state by providing computational proof verifying the current state post state transition function through initial full node synchronization; as full nodes are often required to backtrack all the way to be genesis blocks on a network to be fully assured of the validity of the most current state. While this use of zk-s[nt]arks does have its place, this does not mean that nodes on the network (validators on a PoS based chain like the beacon chain on Eth) will no longer be required to store large data sets which suggest that zk-s[nt]arks may be a technology better suited another task involving augmenting the application layer rather than the architecture layer of a system. In essence, zk-s[nt]arks are not simply limited to use in first layer systems; using snarks to make changes to the application layer as a method of bundling transactions (i.e

rollups) is also a very plausible and even reasonable approach to scale – a subject of interest that will be revisited later.

Zk-s[nt]arks have some very interesting properties and when rolled over into a space where cryptography is adapted these properties translate over into the block chains they are implemented on, but first the fundamentals.

Zk-snarks – The acronym for zero-knowledge succinct non-interactive argument of knowledge

Zk-starks – The acronym for zero-knowledge succinct (scalable) transparent argument of knowledge… zk-starks being the predecessor to zk-snarks.

While the Braincell core devs are far more focused with the implementation of zk-starks (on a case by case basis) over zk-snarks neither should be neglected. Both are still fundamentally founded on the idea of succinctness with the added benefit of zero-knowledge.

The succinctness in a zero-knowledge proof system is actually more difficult to achieve than the zero-knowledgeness. The zero-knowledgeness lies in the math of coding some problem into the correct "form", that of a polynomial. As zk-s[nt]arks themselves cannot be directly applied to computational problems. Rather, zk-s[nt]arks are built by converting the code of some computational problem into a QAP.

The fundamentals of understanding snarks rests on the foundation of understanding the simplification of algebraic expression and an elementary knowledge of the distributive property. Should there exist a set of functions:

$B(x) + C(x) + k = (B + C)(x) + k$, where B and C are polynomials and k is some arbitrary constant.

and an understanding of the distributive property exist where $(P – Q) = (\sqrt{P} - \sqrt{Q})(\sqrt{P} + \sqrt{Q})$. Then converting computational problems into QAPs will occur with relative ease. This is followed by the introduction of some complexity in creating the actual zero-knowledge proof itself, after which a process by which verification of said proof by a separate party (a prover) can occur.

Given the polynomial:

$$F(x) = x^5 – 2x^4 + 10x^3 – 20x^2 + 9x – 18$$

Where all conjugate pairs of non-real, complex roots are ignored and only real numbers are considered valid solutions. (Leaving the only valid zero as 2).

Some party must somehow prove that they are aware of the solution to the polynomial equation without ever knowing what the equation that led to the solution is.

Defining our function:

```
# arbitrary arguments
# def func(argu1, argu2, … arguN):

def foo(x):

    y  = x**5

    return x + y – 18
```

where our programming language is restricted to support only basic arithmetic operations (/, *,– , +) and constant-power roots and exponents [x^(1/5) and x^5 but not x^(1/y) or x^y] and simple variable assignment which allows our function to be complex enough that it could support just about any computation where the number of computational steps is bounded, i.e. loops based operations do not exist.

In the same way that polynomials and exponents can be prime factored, flatting operations covert your original code into sequence-based statements where code defined as variables can be broken into constituent elements. Given the statement above the flattening process could appear as:

```
arb_syn1 = x * x * x * x

y = arb_syn1 * x

arb_syn2 = y + x

output = arb_syn2 – 18
```

It is through such flattening process that gathered input variables and the complex body and highly structured code can be broken into a set of statements that still matches the original code provided.

The fundamental structure provided is then mapped using a defined constraint system. In the same way that an arbitrary polynomial which defines a function, P(x) with at least a single real root, has its interception(s) with the x-axis defined as its real roots or factors; the x-coordinates which define P(x) can used to define constraints. Essentially by evaluating a given polynomial at different x-coordinates we define vectors in the system where these vectors are used as constraints in the creation of multiple logic gates where these logic gates very often number in the thousands in practice. A solution vector(*s*) can then be derived from the assignment of all other established vectors (*a*,*b*,c) in the order which they are mapped is then created where the solution vector *s* satisfies the equation:

$$(s*a)(s*b) – (s*c) = 0$$

Where the solution vector itself is then used to create the conditions for the necessary logic gates and the solution vector multiplied by the last group vector is the equivalent of the product of the solution vector multiplied by both the prior group vectors.

Various polynomials are then created from these vectors identified from the constraint system that satisfy the necessary conditions and it is these polynomials which define the parameters for a QAP. The constraints are then checked simultaneously through a bundle of addition and multiplication of polynomials where the output itself is a polynomial. For all checks to pass the resulting polynomial when evaluated for at each and all x-coordinates and points that do correspond to some logic gate must be the equivalent of zero. After which the program, if deemed valid through checks, is compiled. Where the system functions well when a finite set of integer-based elements are used rather than just simple well-defined real numbers which allows it to function well when paired with ECC based technology and ensure protocols built with the technology are actually secure. In short if the program is compiled into an equation in which polynomials are introduced:

$$f(x)\, g(x) = a(x)\, b(x)$$

in which the results of the polynomial can only result in a binary, Boolean result. The equality will hold on the condition that the program itself has been computed correctly. This is essentially the action that the prover is attempting to convince the verifier.

While this is a rather simplified version of how zero-knowledgeness can be mathematically generated, as mentioned prior it is not even the zero-knowledgeness in the system that is most intriguing in the use of zk-s[nt]arks; rather the most important aspect of zk-s[nt]arks along with several other techniques for scale is succinctness. As the succinctness in a system is what contributes to scale; where long computational works are replaced by significantly smaller messages.

The most common means by which succinctness can be encoded into a system is often by introducing randomness as a feature through some means like random sampling. Should there exist a proof of stake chain such that a validator must verify that some activity, Merkle root, block, transaction etc. is valid possibly presented by an aggregator; they can do so without checking the entirely of said block or bundle of transactions through the selection of random checkpoints. Should the prover be presented some secret checkpoint where some polynomial function P(x) must be verified (since our computational work has been encoded as a polynomial) there may exist randomly selected checkpoints that increase the likely hood that the aggregators computation is valid to do so. And while one cannot be fully convinced that the computational work is entirely valid, by having multiple, randomly generated checkpoints confidence in the computation can be built to near certainty which saves exponential amounts of time and reduces the size of the necessary proof. By also ensuring that functions are homomorphic we are able to evaluate programs on data that has already been encrypted to result in an encryption such that given two messages, the product of these messages post encryption will be the equivalent of the encryption of the product of both messages.

S[nt]arks also require very little to zero interaction and they allow such a property to exist where computational work can be verified where just a single message from prover to verifier is required and where anyone can be a public verifier, and only that single message is necessary to determine the validity of the computation itself without the need for new interactions to occur

besides this message. In essence we can create a system where a verifier is required to know nothing of crucial importance particularly from the data that can later be regarded “toxic waste”, as that could compromise the system in such a way as to allow fake proofs to be constructed, and still verify computational validity. Hence the necessity that the witness string remains unknown.

Given an elementary state transition function (STF):

Where the function is defined as $\mathrm{APPLY}(S, TX) \rightarrow S'$

Or said function is defined in a different manner such that $\mathrm{STF}(S, B) \rightarrow S'$
Where **S** and **S’** are representations of states, **B** is a transaction or block, and **SFT** is the state transition function.

We can evaluate the most fundamental components in a base transaction as a function such that:

f(Ω1, Ω2, *o*, *r*, Mp*o*, Mp*r*, *v*) = 0 as long as Ω1 and Ω2 are variables which represent the root hashes of an account’s Merkle-trees pre and post STF respectively, *o* and *r* being the transaction *originator* and its *recipient* accounts, Mp*o* and Mp*r* are the Merkle-tree proofs that verify the balance of the transaction originator, *o*, is at least value, *v*, pre STF, Ω1, and that the value, *v*, if transferred to the recipient, *r*, from the originator, *o*, and hash outputs the post STF value, Ω2. Such a function can be verified with ease should all variables be given, but in an instance where the function, f, is to be converted into a zk-s[nt]ark there are certain aspects of the function that cannot be public knowledge. For the function, f, to be converted, it is imperative that only Ω1 and Ω2 are public knowledge while all other input values in our function (*o*, *r*, Mp*o*, Mp*r*, *v*) are the unknowns that form the witness string.
In this function it is the zk property that then allows the verifier to confirm that the prover does in fact know a witness or, more likely than not, a set of witnesses (*o*, *r*, Mp*o*, Mp*r*, *v*) that cases the root hash of the input Merkle-tree to move from Ω1 to Ω2 in such a way that the transaction occurs in line with certain requirements. It is the homomorphic multiplicity property that allows for such to occur (where multiplications can be performed on multiple encrypted data sets and still produce the same result if the multiplications were first performed and then the results encrypted). In a vast majority of cases there is often not just a single witness but rather a set of witnesses that fulfills the necessary obligations to serve as proof. Here, the arbitrary size of the witness string itself plays no role in the verification of computation which causes s[nt]arks built from arbitrarily complex problems and those created from simple problem will require the verifier to expand the same effort. Rather the parameters that influence verification full to things like the capped size of inputs.

s[nt]arks have been shown to be particularly useful in solving problems in complexity theory, especially problems of the class non-deterministic polynomial time complete (an example being the graph-three coloring dilemma). QAPs (as discussed earlier) are also contained within the class of problems NP and are even more advantageous due to increased efficiency in situations where the verification process can be succinctly represented by an arithmetic circuit. But like almost all s[nt]arks and s[nt]ark fundamentals; QAPs capture all circumstances in the class NP, they are not NP-complete as QAPs are not a computational language, rather they are a means to

model computation. What is often referred to as NP-complete are the more specific instance of QAPs such as QAP SAT and R1CS (the restraint system) as an arithmetic circuit satisfiability can be effectively reduced to QAP satisfiability. As QAP satisfiability are more expressive they are regarded as a language as opposed to simply being a model.

Quantum computing and Cryptography

While most of cryptography is generally based either on some form of SHA256 or elliptic curve cryptography (ECC). If one were to consider quantum computing, and assume that in the near future quantum computers will be as prominent as ASICs are today then a need arises, one involving the early implementation of new quantum proof alternative measures to improve on existing concepts early on.

Quantum mechanics when implemented often completely alters the fields of classical information theory and especially classical computation which many blockchain protocols rely heavily on.

The current group algorithms that seem to be the most relevant in relation to quantum computing and cryptography seem to be Shor's algo, which is often used to simplify the complexity of mathematically problems. One of these methods of complexity simplification being the use of Shor's algo to perform prime factorization. Shor's algorithm has also been used to break hidden order groups and the prominently used elliptic curve cryptography.

Assuming knowledge of basic modular arithmetic and the set, $\{s\}$, of integers contained in each mod as common knowledge where the set of integers contained is represented as:

$$\textbf{mod } x = \{0, x\text{-}1\} \textbf{ or } |x| = \{0, x\text{-}1\}$$

Now given some equation where $\mathbf{N = pq}$ where both variables $\boldsymbol{p,q}$ are primes and given an integer based input for $\boldsymbol{N}$ primes p and q can be determined through group theory and Euler's totient function.

With Shor's algo factoring can take place in $\boldsymbol{Nq(t)}$ on a quantum computer where **N** is the number of bits of the number which is to be factored, t is time (polynomial time taken in $\boldsymbol{logN}$). In classical computation where numbers are determined in bits {0,1}; quantum computers run qubits allowing for the set {0,1} but also some superposition of 0 and 1.

The number $\boldsymbol{N}$ to be factored is turned into a problem of finding the period of a lengthy sequence. By using a quantum computer as a computational interferometer (where interference refers to the overlapping of waves which create a superposition).

The other relevant algorithm that could affect cryptography is Grover's algorithm, which is essentially an unstructured search that makes it faster – than would be the case using classical compute – to find some variable which represents some number that satisfies a specific property in a function.

Where the total amount of time for this unstructured search using classical compute would be **O(*N*)**, the amount of time necessary with Grover's would be **O(√*N*)** in a list of items ranging from 1 to N where N is the size of the function's domain.

Qubits need to be put through gates to then put them in superpositions:

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle.$$

Here lies a threat that Grover's could present to cryptography and to zk-snarks. Problems within the class of nondeterministic polynomial-time complete (NP-complete) are generally considered to be difficult due to their search spaces having no identifiable structure, and as there is some zk-snark that can be applied to all problems in NP it is possible that Grover's algo could be used to gain quadratic speedup in the unstructured search needed to satisfy some cryptographic key. This is also the reason that the secret randomness that is involved in the creation of the parameters that create a zk-snark must be disposed of along with other toxic waste.

Should a quantum computer be fitted with Grover's for the task of unstructured search as is the case with mining, it is possible that such a computer would gain a quadratic speed advantage over other devices present in the field. Although is possible that the complexities involved in maintaining quantum states along with the many physical qubits (<= 1000) needed to create a single logical qubit that could be error tolerant, the introduction of quantum error correction and the opportunity cost presented by such could far outweigh the benefits of using quantum computers for mining.

The Braincell Core devs are already aware of, and have already begun implementation of quantum proof solutions where the need has risen as the use of these quantum proof solutions often come with efficiently tradeoffs of their own that must be optimized for.

Eliminating the Need for Trust

When implementing zk-snarks the system must rely on a "trusted setup" where toxic waste (which includes the witness string that could allow a prover to create false proofs if know) must be disposed of. While almost every blockchain makes use of some form of honest majority or rational majority of participants it would be of significant benefit if one could eliminate, or in the very least reduce, the need for trust in a system. So, while zk-snarks are a revolutionary solution to scale there do exist scenarios where an even better means to scale is suitable. A means that removes the necessity of a trusted setup, zk-starks.
Where zk-starks are implemented over zk-snarks, trust is replaced by transparency; a highly desirable feature in any blockchain system. Zk-starks are even secure against quantum attacks. The tradeoff that comes with the use of zk-starks as opposed to zk-snarks comes in the form of the increased size of data proofs as they increase significantly from around 288 bytes to several hundred kilobytes. Despite the many benefits of zk-starks there are times when the 100x increase in the size of proof is uneconomical, but this is usually in the context of smaller of use for

smaller applications. In regards to its use in decentralized public blockchains where trust must be minimized, and in the event that the rise of quantum computers come sooner than anticipated, then this tradeoff between increased proof size for increased security will be well warranted.

Just like with snarks, starks can be regarded as a computational game of sorts, one that also take place between a prover and a verifier, where the prover could be any public entity participating in the "game". Here, a public function exists such that:

$$f(x) = y$$

where *f,* is a public function, *x* is a private input, and *y* is a public output. The operation of proving that one has knowledge of what the private function, *x,* is that satisfies the function *f*(*x*) which produces some output, *y* without explicitly stating what *x* is becomes the task required to verify the proof. As discussed in previous sessions the succinctness is more vital than the zero-knowledge, and in creating a proof that is succinct the computation must be done in such a way that the proof is verifiable in a much shorter amount of time than it would take to simply compute the public function *f* itself.

If the function *f* were some computation that would take a time *t1* of 1 month using classical compute on a simple computer, but time *t2* on an application specific device for computation where $t2 = \sqrt{t1}$. While both methods of performing the computation produce the output *y* with a verifiable proof the time element differs drastically between the method by which the computation is performed.

In a blockchain the function *f* could be a function that processes a block and verifies its validity, *x* could be the input of variables such as transactions, and *y* the hash of the block post STF. The output of the function *y* could also be the output of the function after checks have been performed to output a Boolean value, 1 if the checks pass and 0 if they do not after the STF. Given an encrypted transaction such that "*Xs1* is a sender's previous balance pre STF. *Xr1* is the receiver's previous balance pre STF. *Xs2* is the sender's new balance and *Xr2* is the receiver's new balance". These elements could help form the proof necessary which verifies this entire transaction's validity (the transaction avoids double spend and ensures non-negative balances else the all changes revert and confirms that *Xs1 - Xs2 = Xr2 - Xr1*) if *x* is the pair of encryption keys for both the sender and receiver accounts.

The values for *x* are usually quite large in more practical examples, and in such examples the possible values for *x* that could be input into the function should lie outside the constraints of the system to create a possibility to check for values outside of *x* when there exist some set of values that could satisfy the function.

Through STARKs, even in their most basic form the single PoF that is inevitable with the use of a trusted setup as is the case with SNARKs is removed.

More Paradigms for Scale

The concept of scaling in terms of blockchain capabilities and security is not simply limited to the consensus layer and the consensus mechanism a blockchain takes up using. There are several other things that can be done in an attempt to scale and raise the efficiency of a blockchain. The choice lies in the pursuit of expanding functionality in base-layer blockchains themselves or the implementation of clever protocols which rest of top of existing blockchains in the application layer in the same way that Internet architecture, HTTP serves as a part of the application layer resting on top of the TCP/IP in the internet protocol suite.

In that sense, first layer (L1) blockchains are those existing blockchain architectures in which second layer (L2) networks are built on top of. Layer-1 implementations for scale involve making the underlying blockchain itself capable of processing more transactions at a faster rate despite the limiting capacity of participating nodes through the modification of the blockchain protocol or to how clients work on the blockchain. This could involve blocksize increases which tradeoff reading and writing to the chain in favor of the base-layer chain becoming capable of processing more as well as sharding (The subject of scaling via layer-1 which we have already discussed prior in this document during the block size debate). And while there exist different solutions for scale that can all be classified as “Layer-2” solutions, these solutions vary wildly in terms of both implementation and properties, and it is best to avoid such vague grouping terminology when possible but to use the term very broadly in this instance; layer-2 solutions for scale generally encompasses anything considered an application-layer design pattern intended to increase scalability and quite often involve moving complexities off chain while still providing valid proof, an example of such being the implementation of a stateless client, optimistic rollup, ZK-SNARKs / ZK-STARKs or other such layer specific changes as well as Plasma solutions.

Braincell research foundation and core devs intend to further not just itself but the community as a whole through the development and early implementation of layer-2 heavy design solutions for scale that do not replicate the layer-1 scaling mechanisms of the networks and mainnets it is developed on top of through the use of both rollups.

And while Braincell does provide L2 solutions to scale, that is not to say that L1 scalability solutions do not have their place; particularly when it comes to preserving a more immediately legible blockchain and the infrastructure organization around specific solutions. We simply have taken up the stance that involves the freedom of choice for DApps choosing what tradeoffs they would like to accept for immediate scalability and minimizing unnecessary complexities within the consensus layer. Should building certain expansive features be programmed into the base layer blockchain it may facilitate governance disputes to occur amongst opposing parties. For the Braincell core devs as the various projects we are spearheading evolve is it very possible that we may have to venture into the base layer protocol space to be a part of its maturing process and gradual stabilization by building a multiclient capable decentralized network as the Braincell core devs argue that base layer blockchains are yet to reach the minimum level of complexity that is necessary for base layer chains to be powerful enough for applications to build freely on top of them. This increased complexity should be coupled with forward compatibility and unnecessary complexity should also be avoided. It is worth considering that only a handful of features should only be implemented on base layer chains as opposed ton putting then on

protocols which rest on top of the chain's base layer, these features usually being publicly accessible goods such as random number generation for gambling where goods like these are essentially subsidized by block rewards and transaction fees. But even then, it is inevitable that post stabilization of maturity of these base-layer blockchains, layer-2 chains, and solutions will become the primary subject for rapid growth and evolution of blockchains.

It must be noted that while some L1 and L2 implementations of scale do offer essentially the same solutions these solutions *generally* cannot "stack" or build on top of each other to further increase scale (ie if a L1 solution for scale is implemented eg fraud proofs, provides scale by a factor of 100 and a L2 solution for scale is also implemented eg optimistic rollup, and it too provides a factor of scale of 100, the scale is not increased by a factor of 1000 but instead is still increased by a factor of just 100 – the level of scale provided through either solution even if both are implemented. As these solutions for scale, though performed on different layers, still essentially perform use the same mechanism. So, should both solutions be implemented at the same time the result is simply infrastructure bloating rather than increased scale.

But while both L1 and L2 solutions may involve the implementation of similar mechanisms for scale the results can vary wildly. So much so at times that it may make the solution for scale on layer-1 so drastically limited that it would only be reasonable if implemented on a different layer. An example of such a case being the use of *fraud proofs* on a network.
When fraud proof scaling techniques are proposed as a form of improvement to how clients verify the state (Layer-1); they do so as a mechanism intended to secure a vulnerability introduced by attempts to scale using Simplified payment verification (SPV). Light clients which do not fully validate blocks (as they not to verify the current state) introduce a vulnerability creating the possibility that these clients could be fed false or invalid block information from malicious full-nodes on the network; this emerges as a result of these clients not processing every transaction in a block (ie not fully validating the block) and rather *trusting* that the proposed block is correct until proven otherwise.

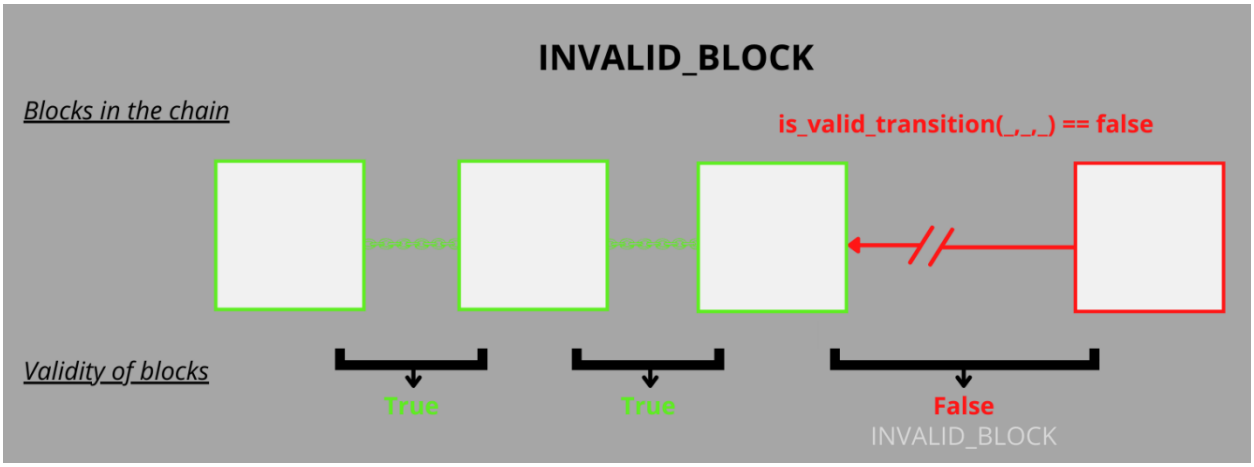
While this client efficiency improvement does work, this L1 fraud proof mechanic does have its limitations as state and data consensus must occur as separate processes; else every node proposing a new block would need to verify all recent blocks itself prior to publishing a block of their own. A process that would make scalability gains negligible.

Even though these clients do not verify the state they do still download all blocks to check for availability; but, these clients with accept blocks by default and only reject them after receiving a Merkle proof through the network proving the invalidity of said block.

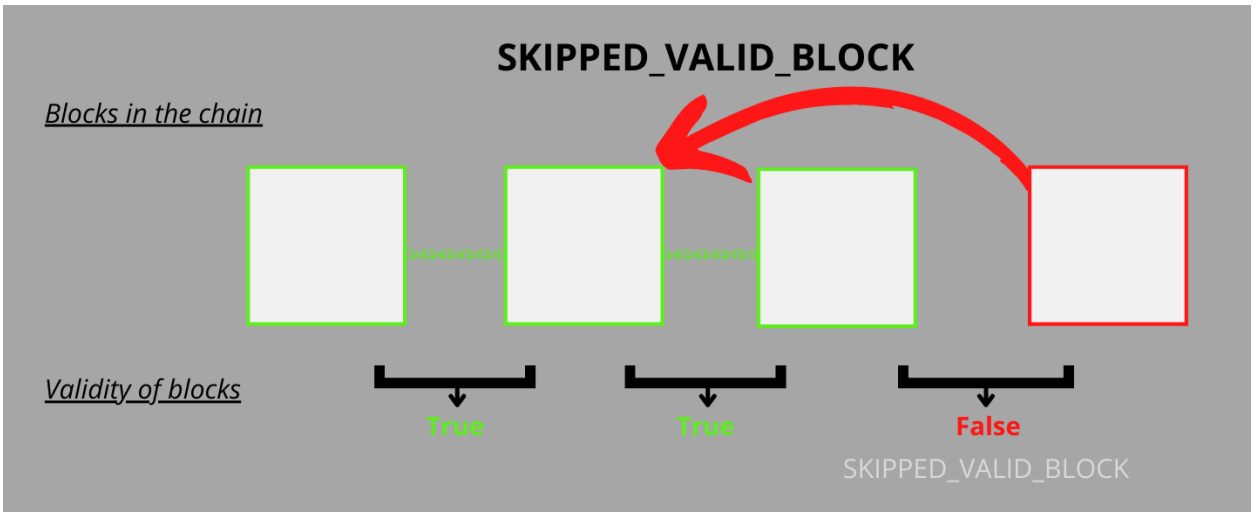
An alternative, the use of optimistic rollups – which functions by having a system that stores a series of historically verifiable state roots, but does not officially "verify" or produce a state of finality until sometime after the challenge game period ends (which is usually around 1 week, but in several cases the challenge game period can be set by whoever calls the function to start the task itself. Here the responsibility of setting a timeframe for the challenge period falls on the rollup creator) before the newly added state is finalized. Optimistic rollups exist in rollup style contracts on the Ethereum mainnet where the fundamental purpose of these rollup style EVM

contracts is to create tuples consisting of transaction data for sequencers or aggregators to process off-chain before submitting (without any computational proof) batches of highly compressed transaction data sets to the main chain that it is secured by, where these batches of transaction data sets have been striped to contain the minimum amount of data required to verify the validity of the transactions and update the state having been "rolled up" or bundled by the sequencer. This aggregator – who could be selected to play this role in several ways not exclusive to random sampling and rotation amongst stakers who met certain requirements such as bond deposits – serves as the central party tasked with compression of transaction data; computes the new state root after the bundled transactions are applied and then submits their newly proposed state through calldata or a callop function *add_block* as an optimistic rollup block (where this optimistic rollup block has enough space for its validity to be disputed while a challenge game period is still active by disputing the validity of the proposed lookup table) with the 'assumption' or 'hope' or 'optimistic view' that the submission is or will be valid. . Anyone can download the proposed block and attempt to verify its validity through the state transition function with a proof. The ongoing potential bounty on each aggregator and the assumption of an honest or rational majority are security measures intended to ensure the propagation of honest blocks without proof. The aggregator's bounty can be won should any of a small set of conditions that show ample evidence of an invalid proposed block be met:

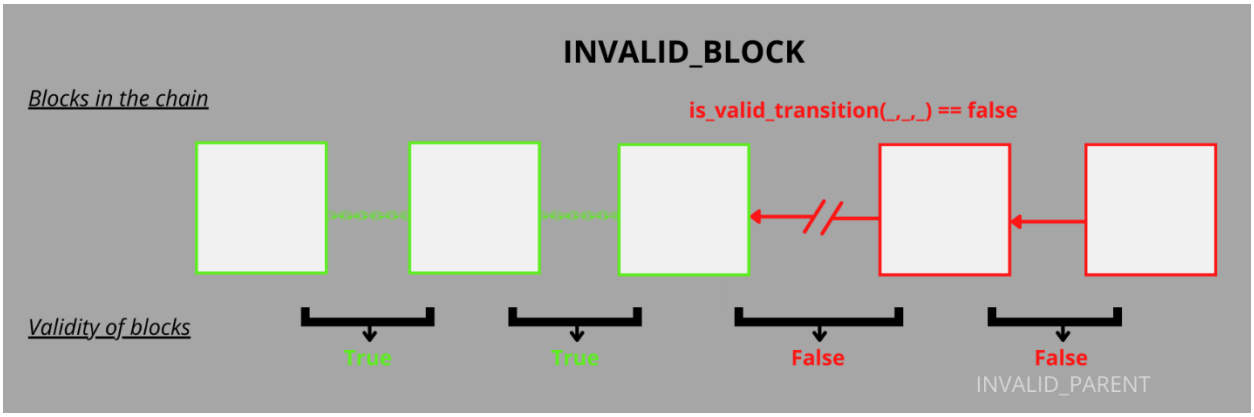
- A full node proves the invalidity of a committed block with the opcode '*is_valid_transition(prev_state, block, witeness) => boolean'*



- A full node proves that the proposed block has "skipped" a valid block.



- A full node proves the invalidity of the block that the proposed block was built on.



Essentially the parent block to the aggregators proposed block can also be proved invalid, which would require aggregators to also verify the validity of the previous block before building on top of it. Should a full node discover an invalid while the challenge game period is still open it would not simply challenge that block (as that block could also have an untrusted pre-state), but backtrack all the way to the very first invalid block and challenge that block; where the potential for not just a portion of a single aggregators stake could be won, but the stakes of all aggregators who built on top of the first invalid block while mitigating the risk of making an invalid block or challenging a block originating from a untrusted pre-state.

Should a watchtower full node discover an invalidly proposed block through the challenge game the disagreement will trigger the on-chain execution of all transactions in the package proposed by the aggregator. If the aggregator's proposed state changes are proven to be invalid his bonds are slashed and a portion of those bonds are rewarded to the watchtower node that discovered the invalid block and the state reverts to a point in time prior to the state proposed by the malicious aggregator.

What may be the biggest drawback involved in the use of Optimistic rollups is the time associated with the challenge game period itself. The frequency of emerging nodes which take

up these challenges is also very low and with the risk of having a deposit slashed, disagreements are very sparce. Also, during that period in which a challenge game window is still open coins locked into the system cannot be withdraw until the challenge game period itself (usually around 1 week) ends. Which limits the use of optimistic rollups to things outside of simple payments.

Decentralized oracle networks

Oracle mechanisms serve as a decentralized computational system where the intention is to reach a state of consensus. And while blockchains in general are perceived to have the same intention, the difference here lies in the area of focus which decentralized consensus is intended. Oracles are built for the purpose of reaching consensus on an event whether present or past based on some data source(s). Decentralized oracles in the blockchain space can be used to create a "metalayer" between the non-deterministic highly unstructured world that exist outside of blockchains and the reliable, proof driven world that is blockchains in such a way that unstructured data can pass through this "metalayer" where consensus can be reached on the information; after which this information can be relied upon in the structured, tamper proof world of blockchains.

Oracles are not based in objective truth, but rather definitive truth where in some number of nodes or data sources can come to a consensus using some particular method with some specified threshold of agreement. Should these conditions be met, it is possible for an oracle to reach definitive truth. This consensus-based truth is far more desirable than the alternative of a single entity or data source defining truth.

The use case for oracles, lies in their ability to allow smart contracts to interact with external data sources and real-world data that have been confirmed accurate through consensus by oracle nodes.

If a farmer were to deposit a defined amount of funds into a financial derivatives smart contract that pays out inversely based on precipitation levels in his region, then this contract must be able to interact with some external data source to accurately determine how much rain fell during the period for which the contract coverage as defined by the contract exist. The smart contract cannot simply rely on some centralized feed, as there does exist the possibility that the feed is presenting fictious data. But should there exist an oracle that gathers data from several (e.g. 100) weather feeds and then takes the data provided by those sources up to the 60^{th} percentile and averages that value then it is much more likely that the oracle's determined precipitation level is an accurate definition of reality. This oracle determined precipitation level can then be fed to the smart contract which can initiate payouts automatically to the insured farmer if there is not enough rain based on the defined agreement coded into the smart contract.
This crop insurance example being the most well-known use of the combination of oracle contracts to feed external data to smart contracts which then activate based on the information received by the oracle and the contractual agreement coded into the smart contract.

Smart contracts are made highly reliable and secure in such a way that they can only communicate with highly reliable information sources that are within the defined system, essentially tokens and private key info. Smart contracts cannot simply retrieve information from an API. Should this be possible the data retrieved from could create security concerns for the blockchain itself and several consensus issues on how agreements can be reached on the information provided by the API. With the use of highly reliable on-chain code that is well defined by some contractual agreement and is dependent on some decentralized oracle mechanism to confirm some event, series of events, price data, etc where the reliable external data sources provided to the oracle can be considered definitive truth through consensus of oracle nodes and decentralization conditions which are agreed upon as sufficient to control our on-chain smart contract then new horizons for automation and new transaction types in blockchains can be achieved through this tamper proof on-chain contract controlled by a mutually agreed upon proof system or trigger.

Oracle contracts are almost always built for specific use cases. There exist oracle nodes that can generate randomness for on-chain use in public goods such as the generation of randomness in a random number generator for on-chain gambling.

Should there exist some oracle network, the network itself must be capable of expanding the number of oracle nodes relative to the value secured by the contracts on the network. In a relatively small oracle network where the amount of value secured by contracts is only 1 ETH then it is reasonable to have 10 oracle nodes and 4 reliable data sources to secure that value, but should the network have a value secured by its smart contracts be valued in the 100s of ETH then it would be unreasonable to use the same amount of oracle nodes and data sources to secure that value. As such the oracle network built or used must be capable of expanding the number of nodes on the network relative to the value secured by the network and in turn by the smart contract.

The Braincell core developers are aware that there are several groups in the blockchain space that may have developed their understanding of the blockchain space through things like non-fungible tokens (NFT)s, and may not have as much experience with various tokens or staking mechanisms. As NFTs cannot be defined as liquid assets by any means it may limit participation in on-chain activities, but with oracle networks even illiquid NFT users may be able to stake their NFT and participate in on-chain activities in the Braincell network.

Optimistic Rollups and Staking Mechanisms

As mentioned prior, optimistic rollups (more specifically optimistic rollup style blocks) function by storing a series of historically verifiable state roots, without finalizing the state changes until the challenge game period of usually one week comes to a close. Just like all rollups they are intended to create bundles consisting of transaction data for an aggregator to process off-chain before submitting a highly compressed version of the data and outputting the new state after these transactions have been applied without submitting any computational proofs to verify the validity of the performed computation. As aggregators are required to front some “stake” in the

optimistic rollup to the smart contract as form of bounty, it is possible for this stake to be something other than just the token native to the network. As long as whatever is proposed by the aggregator as “stake” is coded to be accepted by the smart contract. This could even include illiquid assets which could be staked as collateral. While most Illiquid assets cannot earn their owner’s tokens in most blockchains and usually sit dormant in the owner’s EOA until sold to someone else; it is possible for these illiquid assets –such as NFTs – to be staked in an optimistic rollup style contract by their owners allowing them to participate in a validation of transactions and state changes and even giving them the potential to earn rewards while using an illiquid asset as collateral on the network so long as the feature to stake NFTs are coded into the smart contract. If such is done, it is possible for these NFTs to be staked by their owners as the collateral necessary to participate in the challenge game involved in optimistic rollups. The aggregator or disagreeing verifier of a particular block could theoretically stake their NFTs via smart contract and should a dispute arise (an event that rarely happens with optimistic rollups as the consequences are always costly) either the aggregator or disagreeing verifier be proven to have proposed an invalid block or invalid proof respectively their staked NFTs would be slashed respectively and provided to the opposing party in the challenge game.

Coded into all rollups there is a generally a static minimum number of tokens required by the smart contract for a node to participate as an aggregator/sequencer (if they are randomly selected to do so) or a validator of the proposed block. To create a system where NFTs could be presented as “stake”, their value must be determined through some means to be the equivalent of or more than the value of the minimum requirement of tokens to be staked when attempting to participate in the propagation of the rollup style block. These values for potentially staked NFTs can only be determined through a decentralized oracle mechanism. The external data regarding the value of the NFT cannot be determined simply by the last sale price or the floor price as these are not the definitive values as determined through decentralized consensus. Rather an oracle must specifically be used to provide decentralized values for specific NFTs potentially provided. One can take an oracle network and even place a portion of computation done into the oracle network, so long as participants in the network are confident in the set of oracle nodes being trusted users who could have some knowledge of the computation itself. But even with that, through the use of trusted execution environments that these oracle nodes can be forced to run this would allow for even the committee of oracle nodes to be left in the dark as to what computation took place, even through backtracking the process, while the users are still guaranteed the outcome of the computation.

The foundation is determining the best means to structure such an oracle but it is very likely that we will rely on a combination of random sampling and providing block rewards to those up to the 66^{th} percentile as users who desire to earn rewards are incentivized to provide the same answer as every other participant, that which is the closest value to the definitive truth. With the ever-expanding art projects in the space, it may not be feasible to value every piece of work through an oracle system, therefore staking may be limited to more well-known NFTs as we slowly expand. We can use this data to then define definitive truth for our oracle and then have our smart contract use this external data to value NFTs to be staked. These optimistic rollup style EVM smart contracts can hold not only tokens but other assets and it is also possible to issue

tokens based on the oracle's determined value of the asset. After the challenge game period concludes these smart contracts can return tokens or other staked assets back to the private keys from which they originated. Simply because there exist core developers that may not have an understanding of why the uniqueness of a piece of art carries some arbitrary value in blockchains, the community consensus says that a value does exist and therefore a use will inevitably arise out of that. Also, even if the value of an NFTs – as determined by the decentralized oracle – does not suffice to meet staking requirements multiple NFTs may be staked as collateral if a single one is determined as insufficient in meeting the minimum value requirements.

As the biggest drawback in the use of Optimistic rollups seems to be the challenge game period itself; should illiquid assets such as NFTs be staked they will also be held by the contract until the challenge game period expires. But through staking NFTs rather than liquid assets, like tokens, which would have likely remained dormant in storage otherwise we have introduced a use case for what would normally be an unusable asset until sale.

And while there will also be a process for making NFT based staking terms for smart contracts that process is much simpler than mapping real world financial contracts with ambiguous statements to smart contracts.

Scalability – ZK-Rollups

While the optimistic rollup use cases cannot be overlooked, they are not well suited for activities like simple payments as funds or assets deposited in the optimistic rollup cannot be withdrawn until the challenge game period ends. A system meant specifically for simple payments that ensures the safety of funds for immediate withdrawal is more plausible using zk-rollup architecture. The best way to do so is with the implementation of a rollup that is also based in zk-snark technology, a zk-rollup.

In several iterations of first layer blockchains all nodes in the network are required to process all transactions published on chain specifically those which still rely on PoW as the method of achieving consensus. Here, miners are the entities responsible for including transactions, but regardless of the transactions the miners add these transactions are still processed by every node that downloads and validates the block and not just the miner who has chosen to include the transaction. Transactions are still processed by every node that downloads and validates the block and not just the miner who has chosen to include the transaction. The same is the case for contract code, it is also executed by all nodes that validate the block, but just like in an optimistic rollup (and with rollups in general) and with EVM style rollup contracts, permissionless transmitted data like transactions are compiled for aggregators to process off-chain. The aggregator/sequencer is then tasked with striping the set of transactions to only contain the minimum amount of data required to verify the validity of the transactions and update the state, all other data is left out. This striped data that is actually necessary in updating the state is then heavily compressed or "rolled up" by the sequencer and then published as a blob on-chain. The new state that would result from the transactions provided is then computed off-chain. In the case

where the ideal rollup for transactions is used, the zk-rollup, the computation is done off-chain just as with all rollups and the proposed block is submitted with a zk-snark that provides proof that some arbitrary hash of the resulting computation is correct. That zk-snark is submitted on chain with the proposed block and instead of nodes having to perform the actual computations that lead to the resulting block they would only need to very the proof provided by the zk-snark in such a way that they would not even need to know how that proof was generated. While zk-rollups may not be used as prominently as optimistic rollups (because of their ease of use and practicality in the field of general purpose EVM computations) they still win out in all scenarios involving application specific uses and things like simple payments. With the exponential improvement of zk-snark technology that is taking place, zk-rollups will very likely become the most prominently used rollup style defined in smart contracts.
The benefits that come with the use of rollups for security far outweigh any other possible alternatives at the moment. An EVM style zk-rollup on the Ethereum network would inherit the security benefits of the base chain itself. In the worst-case scenario staked funds can still be retrieved even through transacting on the main chain and will never be permanently lost.

The expansive use of these smart contracts will allow for complete transparency and trust minimization in a blockchain format that is adaptable for scale and core conditions are well understood and defined by smart contracts.

Updates to the Network and Protocols

Base on the network architecture of Braincell; protocol updates on the network that take place after being deemed necessary through on-chain voting will occur through bilateral hard forks with replay protection. The reason being developer flexibility, as the new rules will not be obligated to fit into the old set of rules which is a much-needed feature in a cryptocurrency with the desire to assimilate new ideas into the network. Clients will be given ample time to upgrade their clients and will (as hard forks suggest) be given the chose rather than forced to upgrade their clients to continue participating in the network. Decisions regarding forks will generally be put through an on-chain vote where votes can also be delegated.

Tokenomics and Distribution

The cryptographically secured, native, digital token that the Braincell Network and ecosystem will rely on is the Braincell Token. Braincell Token with be issued as an ERC-20 standard compliant digital token on the Ethereum blockchain as the Ethereum blockchain is the initial network where Braincell will demonstrate its scalability and use cases. The Braincell token will function as a unit of payment and a means of staking for those who participate in the Braincell ecosystem.

The Braincell token in no way provides any rights, or interest in the body which governs the network, nor does it present shareholding rights or title to the governing body, its affiliates or any

company or enterprise or body. The Braincell token does not entitle token holders or purchases to dividends, profit, revenue or investment returns, and is not intended to constitute an investment. Ownership of the Braincell token carries with it no rights besides those that may be afforded by the Braincell network or third parties that make use of the Tokens.

As the PoS consensus mechanism requires that participants have some incentive for maintaining the ecosystem on the network and contributing, the Braincell token will serve as the provided incentive for doing so. Participants will be rewards according to their level of contribution as the Braincell Network relies on nodes to consume computational resources to participate in activities such as publishing and verifying snark proofs, and validating blocks. Participants who consume resources to maintain the network are rewarded with Braincell tokens (activities which essentially define mining on the network) for doing so. Braincell tokens will be used to quantify and pay according to the cost consumed in computational resources. It is only users who have contributed to maintaining the network that will receive tokens as an incentive for doing so and to continue doing so. Users who do not actively participate in the network or even holders of the Braincell token who do no actively participate in maintaining the network will receive no Braincell tokens as rewards. As participation in the consensus process depends on one's staked Braincell tokens this serves as a deterring mechanism for malicious behavior and various offenses as nodes are required to stake tokens prior to participating in the ecosystem. Braincell tokens will be slashed in the event that some malicious behavior is committed by the staker of the Braincell tokens during the consensus process.

Braincell tokens are non-refundable and will not be accepted in exchange for cash (or its equivalent value in any other digital currency) by the Governing body, its affiliates or the issuer. Braincell tokens do not represent a means for any token holder to develop any form of rights with respect to the founders or Governing body, its affiliate, or the issuer this includes the lack of rights or entitlement to future dividend, revenue, ownership rights securities, voting, or any form of intellectual property or licensing rights or any financial equities or the equivalent.
Braincell tokens are not a loan in any form to the Governing body, its affiliates or the issuer and they are not a representation of any form of debt owed by the Governing body its affiliates, or the issuer and there is no expectation of profit.
Braincell tokens do no provide the token purchaser or holder with any form of ownership or other interest in the Governing body, its affiliates or the issuer.

Token will be distributed in a manner where a relatively large number of economic resources are not consumed in the process of obtaining these coins to avoid the increasing likelihood that the network encounters a 51% attack. We will also implement an initial lockup and distribution period of tokens as well as monitor the holdings of network participants who are also developers.

Towards a Balance

The initial distribution of tokens by accepting multiple currencies at a fixed exchange rate has wreaked havoc in the past for those involved, so we will be avoiding this approach entirely. Rather, tokens will be initially distributed through the exchange of a single existing

token – in this case that token being that used by the first layer system that Braincell (a second layer system) is built on top of – Eth.

Since the total supply of Braincell Token will be capped at (1,337,069,421 Neura), a completely fixed supply, then because of this fixed supply we expect the price of Braincell tokens to be volatile as the change in demand will be entirely expressed by a change in value of the asset. Half the total possible supply of coins will be distributed initially to allow would be miners to stake their coins to mine both new tokens and receive miner fees. Miners, in such a case, would have an illiquid financial incentive in Braincell's rise. A portion of transaction fees, 9.6% will be burned with each transaction so that the supply rate still tends to zero over time. A portion of funds received through the initial token sale will be donated towards various Ethereum Foundation projection to further the development of the ecosystem while the Braincell team will hold a fifth of the total existing supply for a fixed period (1 years unlock period); after which periodic distributions of the teams' holdings will occur until the team is unable to hold more than 10% which will be used to preserve and fund new projects in the ecosystem.

After which staking will be used to mine the remaining fixed supply of tokens. This supply distribution percentage was also chosen to prevent 51% attacks on the network after its initial token distribution. While this may seem like a centralized form of distribution it is to ensure the network is not disrupted immediately after inception by malicious actors. And as the blockchain trilemma still holds true even more so for early-stage networks utilizing PoS Decentralization for consensus may need to be somewhat sacrificed in the beginning of a network's existence for the sake of both security and scalability.

Braincell Wallet

The Braincell core developers are currently building a user interface friendly mobile wallet that will be integrated with WalletConnect to ensure secure of storage of both Braincell tokens and several other tokens, keys, and single click access to the Braincell Network and a smooth and seamless mechanism that will allow for the Braincell wallet to connect with DApps (mobile and browser based) to the mobile app. Mobile users will be able to smoothly interact with DApps on browsers and even other devices in the near future while ensuring that their keys and tokens are safely stored in the mobile wallet.

Team

- Don Starke. Cereal#1929 Co-founder
- ---
- ---
- ---
- ---
- ---