Orch Network: A Decentralized Secure Multicomputation Platform for Scalable Cyber Contracts, Decentralized Software Engineering and Realtime Dapps

Version: 0.03
Ren Timer *Email*: *ren.timer@orch.network*
May 31, 2018

Abstract.

Orch Network is a privacy-preserving decentralized zero-knowledge secure multicomputation platform for writing, testing and deploying mathematically verifiable secure cyber contracts (smart contracts), decentralized development of software and realtime decentralized applications that behave exactly as intended.

Unlike Ethereum smart contracts with a history of repeated hacks and stolen funds, hundreds and millions of dollars, Orch ensures that no run time errors or intentional/unintentional bugs can be used as exploits to steal funds and corrupt Orch cyber contracts.

Orch uses Simplicity, a new typed, combinator-based, functional language without loops and recursion, designed to be used for crypto-currencies and blockchain applications as its target language and Crack(rather a Turing Incomplete interpolation of Crack codenamed Crackcity) as its source. Crack is a language similar to C/C++/Java that compiles directly from a script to machine code on the fly. Simplicity was released recently by Blockstream, an influential decentralized application development company.

Owing to its Turing incompleteness, Simplicity is amenable to static analysis that can be used to derive upper bounds on the computational resources needed, prior to execution. While Turing incomplete, Simplicity can express any finitary function, which we trust is enough to build useful "smart contracts" for blockchain and blockchain-free decentralized applications.

This work:

(a) Introduces the Orch Network which is a made up of 7 core services. The 1st core service known as Star.Cash is a self-evolving Realtime Unforkable Blockchain Federation with built-in quantum-safe infinitely-scalable(downward scalability) monetary system. Star.Cash doubles up as an asset value protocol for secondary cryptoassets and tokens such as derivatives, utility tokens/appcoins.

(b) Formalizes decentralized Secure Cyber Contract (SCC) structures and their properties, then constructs Orch Network as a Decentralized Secure Smart Contract plus Dapp Platform with secure scalable multicomputation features.

(c) Discusses use cases, connections to other systems, and how to use the protocol.

Note: Orch Network is a work in progress. Active research is under way, and new versions of this paper will appear at orch.network.Forcomments and suggestions, contact us at info@orch.network or send us an encrypted P2P message at our Bitmessage address:BM-NBqZhDGg7vGFCFebhRhT48jNTr8nDbBD

Contents

SL.NO	TITLE	PAGE NO
1	Introduction	3
1.1	Core Services	3
1.2	Paper Organization	5
2	Definition of a Secure Cyber Contract	6
2.1	Turing Incompleteness	7
2.2	Fallacy and Pitfalls of using a Turing Complete language	7
2.3	Properties	9
2.4	Securities And Risk Management	9
3	Computational Boundedness and Decidable Logic	10
3.1	Motivation	10
4	Realtime Unforkable Blockchain Federation and Infinitely-scalable Monetary System	11
5	Decentralized Applications and Use Cases of Orch	14
6	Runtimer: A Decentralized Global Code Repository, Compiler and Runtime System	17
7	DexOS and Vader: Decentralized censor-proof value exchange and transfers	19
8	CrackMachines and Crackle	19
9	OrhOr	19
10	Degents	20
11	Integration with other systems	21
12	Future Work	21
13	Token Units and Policies	22
13.1	Token Supply & Monetary Policy	22
14	References	24

1. Introduction

DApp basically stands for decentralized applications. The first DApp was in fact the Bitcoin itself. And these blockchains and other decentralized open ledger technologies relies on smart contracts i.e. predefined protocols that allow large, ad-hoc, groups of users to transfer value between themselves without needing to trust each other or any central authority. Bitcoin uses Bitcoin Script to create such smart contracts and Ethereum uses EVM with solidity to create such smart contracts. Both Bitcoin script and EVM are much different from each other. Bitcoin script is Turing incomplete thus less complex but less powerful although enough powerful to create a reliable smart contract whereas the EVM is Turing complete thus more complex and more powerful to create more secure smart contracts if only used well. Meanwhile Ethereum is one of the leading open software platform based on blockchain technology that enables developers to build and deploy decentralized applications.

Ethereum can also be used to build Decentralized Autonomous Organizations (DAOs). A DAO is fully autonomous, decentralized organization with no single leader. DAOs are run by programming code, on a collection of smart contracts written on the Ethereum blockchain. The code is designed to replace the rules and structure of a traditional organization, eliminating the need for people and centralized control. A DAO is owned by everyone who purchases tokens, but instead of each token equating to equity shares & ownership, tokens act as contributions that give people voting rights.
But in 2016 ‘The DAO’ project got hacked and since then Ethereum blockchain was forced split itself and their so called smart contracts are less trusted.

Our goal is to create a decentralized platform that enables which can be more reliable and easy to understand and create cyber/smart contracts and decentralized applications as well as deploy these cyber contracts and decentralized applications on a hyperfast realtime unforkable blockchain with integrated monetary system and asset value protocol under active development codenamed StarCash. StarCash is one of seven core services of Orch Network.

Orch Network is a decentralized zero-knowledge multicomputation platform for writing, testing and deploying mathematically verifiable secure cyber contracts (SCCs) and high performance infinitely scalable realtime/near-realtime decentralized applications. One can write SSCs in a Turing Incomplete subset of Crack language codenamed Crackcity that behave exactly as intended.

1.1 Core Services

There are noticeable common features of DApps:

- **Open Source**. Ideally, it should be governed by autonomy and all changes must be decided by the consensus, or a majority, of its users. Its code base should be available for scrutiny.
- **Decentralized**. All records of the application’s operation must be stored on a public and decentralized blockchain to avoid pitfalls of centralization.
- **Incentivized**. Validators of the blockchain should be incentivized by rewarding them accordingly with cryptographic tokens.

- **Protocol.** The application community must agree on a cryptographic algorithm to show proof of value. For example, Bitcoin uses Proof of Work (pow) and Ethereum is currently using PoW with plans for a hybrid PoW/Proof of Stake (PoS) in the future.

Orch uses Simplicity, a new typed, combinator-based, functional language without loops and recursion, designed to be used for crypto-currencies and blockchain applications as its target language and Crack as the source language. Simplicity was released recently by Blockstream, an influential decentralized application development company.

Owing to its Turing Incompleteness, Simplicity is amenable to static analysis that can be used to derive upper bounds on the computational resources needed, prior to execution. While Turing incomplete, Simplicity can express any finitary function, which we believe is enough to build useful “smart/cyber contracts” for blockchain and blockchainfree decentralized applications (Dapps).

Orch.Network’s fundamental building-blocks are its 7 core services and corresponding functional components:

- Star.Cash- Self-evolving Realtime Unforkable Blockchain Federation with built-in quantum-safe infinitely-scalable (downward scalability) monetary system. Star.Cash doubles up as an asset value protocol for secondary cryptoassets and tokens such as derivatives, utility tokens/appcoins.

- OrchOr- Quantum-safe Scalable Zero-knowledge Multicomputation Protocol with integrated zero-proof decentralized voting and transference of proof of zero-knowledge systems

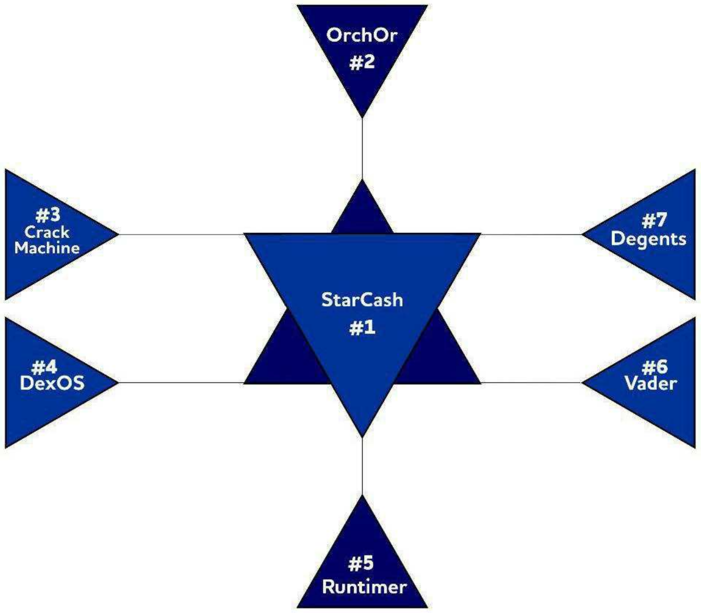
- Crack Machines-Blockchain VMs for running Turing Incomplete cyber contracts in Crackcity language(Crack is the source and Simplicity is the target language)

- DexOS- Decentralized Exchange Protocols for ultra-high speed trading of values and/or cryptotokens/cryptocurrencies without the involvement of any centralized parties

- Runtimer- Decentralized Global Program Compiler and Distribution/Search Engine for software libraries, programs, protocols, blockchains, APIs etc.

- Vader- The interface for implementing drivechains into various external blockchains and blockchain-free decentralized and centralized networks for seamlessly moving data and values into Orch

- Degents- Degents or Decision Agents are strong autonomous hierarchical strong mobile agents acting on behalf of a blockchain/decentralized service to find and map real-world events to facts/conditions/truth values of cyber contracts. Degents incorporates several machine-learning and pattern recognition models endowing it with higher-ordered self-learning and self-planning capabilities. Degents can hop from one P2P to host to another with its execution state and data.



Core Services of Orch Network

1.2 Paper Organization

In the report basically we are talking about Orch Network. Starting from the basic definitions of secure cyber contracts or SCCs and how they can be used in a decentralized network, to what are the other platforms to create smart contracts like Ethereum. Then we talked about the basic component to write cyber contracts which is a language and the basic property of the language which is Turing completeness.

While Ethereum uses Turing completeness language to create its smart contracts Orch uses Turing incompleteness language to create its cyber contracts. Then we moved to the properties possessed by the cyber contracts, and securities involved in cyber contracts and its management.

In the next section we are talking about why we need another decentralized platform for smart contracts if we have a Turing complete language based Ethereum. Then it moves to forkability and how can we avoid it by implementing an Algorand-like real-time unforkable blockchain StarCash. Next section is about running Cyber contracts and decentralized applications. There are also the uses cases of Scalable multicomputation in real life.

The next topic in our report is core services of Orch Network and the dapps enabled by Orch platform.

2. Definition of a Secure Cyber Contract

Smart contracts [1] help you exchange money, property, shares, or anything of value in a transparent, conflict-free way while avoiding the services of a middleman. Basically it is a computer program that directly controls the transfer of values between the network without needing to trust each other or any central authority .

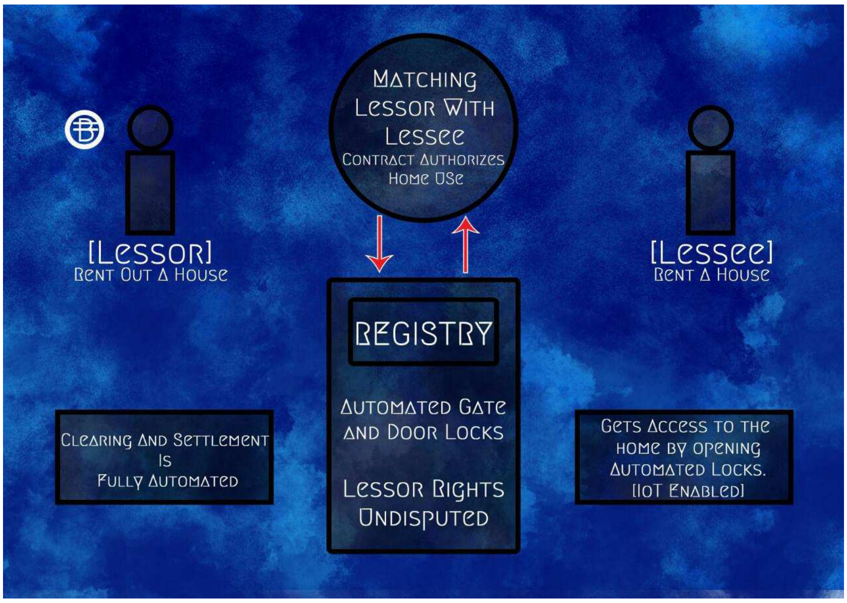
The best way to describe smart contracts is to compare the technology to a vending machine. Ordinarily, you would go to a lawyer or a notary, pay them, and wait while you get the document. With smart contracts, you simply drop a Bitcoin into the vending machine (i.e. ledger), and your escrow, driver’s license, or whatever drops into your account. More so, smart contracts not only define the rules and penalties around an agreement in the same way that a traditional contract does, but also automatically enforce those obligations.

Smart contracts, meanwhile, work hand-in-hand with block chain technology and have the potential to automate and also disrupt processes in many industries.

Whereas a traditional legal contract defines the rules around an agreement between multiple people or parties, smart contracts go a step further and actually enforce those rules by controlling the transfer of currency or assets under specific conditions. In a smart contract approach, an asset or currency is transferred into a program and the program runs this code and at some point it automatically validates a condition and it automatically determines whether the asset should go to one person or back to the other person, or whether it should be immediately refunded to the person who sent it.

A smart contract or cyber contract not only defines the rules and penalties around an agreement in the same way that a traditional contract does, but it can also automatically enforce those obligations.

Smart contracts guarantee a very, very specific set of outcomes. There's never any confusion and there's never any need for litigation. It's simply a very limited, computer-guaranteed set of outcomes.


Matching Lessor With Lessee
Contract Authorizes Home Use
[Lessor]
Rent Out A House
[Lessee]
Rent A House
Registry
Automated Gate and Door Locks
Lessor Rights Undisputed
Clearing And Settlement is Fully Automated
Gets Access to the home by opening automated locks. [IoT Enabled]

This is how a cyber contract works

2.1 Turing Incompleteness

Every decentralized application needs a language to write smart contracts. The Bitcoin uses Bitcoin script to write its smart contract. Bitcoin script is a Turing incomplete language. Whereas Ethereum uses EVM (which is a Turing complete language) and solidity to write its smart contracts. Orch uses simplicity as its core language to write its smart contracts. Simplicity is a Turing incomplete language.

A Turing-complete system is called Turing equivalent if every function it can compute is also Turing computable; i.e., it computes precisely the same class of functions as do Turing machines. Alternatively, a Turing-equivalent system is one that can simulate, and be simulated by, a universal Turing machine.

There are pros and cons of a language being Turing completeness. A Turing complete language is more expressive having loops and jump statements which also make it difficult to use and the programs written in a Turing Complete language are more complex. Another disadvantage of having a Turing complete language is, it is not possible to calculate computational effort before the execution of the program and you need to put a limiter to avoid infinite loops in the program like gas in Ethereum. If a programming language doesn’t have any jump or looping statements, then that language is called Turing incomplete. A Turing incomplete language’s program is amenable to static analysis i.e. an upper bound for that program can be calculated before its execution and allows to place the limits on the amount of computation a transaction can have.

2.2 Fallacy and Pitfalls of using a Turing Complete language

1) Turing-complete languages are fundamentally inappropriate for writing "smart contracts" - because such languages are inherently undecidable, which makes it impossible to know what a "smart contract" will do before running it.

(2) We should learn from Wall Street's existing DSLs (domain-specific languages) for financial products and smart contracts, based on declarative and functional languages such as Ocaml and Haskell - instead of doing what the Web 2.0 "programmers" behind Solidity did, and what Peter Todd is also apparently embarking upon: ie, ignoring the lessons that Wall Street has already learned, and "reinventing the wheel", using less-suitable languages such as C++ and JavaScript-like languages (Solidity), simply because they seem "easier" for the "masses" to use.

(3) We should also consider using specification languages (to say what a contract does) along with implementation languages (saying how it should do it) - because specifications are higher-level and easier for people to read than implementations which are lower-level meant for machines to run - and also because ecosystems of specification/implementation language pairs (such as Coq/Ocaml) support formal reasoning and verification tools which could be used to mathematically prove that a smart contract's implementation is "correct" (ie, it satisfies its specification) before even running it.

Turing-complete languages lead to "undecidable" programs (ie, you cannot figure out what you do until after you run them)

One hint: recall that Gödel's incompleteness theorem proved that any mathematical system which is (Turing)-complete, must also be inconsistent incomplete [hat tip] - that is, in any such system, it must be possible to formulate propositions which are undecidable within that system.

This is related to things like the Halting Problem.

And by the way, Ethereum's concept of "gas" is not a real solution to the Halting Problem: Yes, running out of "gas" means that the machine will "stop" eventually, but this naïve approach does not overcome the more fundamental problems regarding undecidability of programs written using a Turing-complete language.

The take-away is that:

When using any Turing-complete language, it will always be possible for someone (eg, the DAO hacker, or some crook like Bernie Madoff, or some well-meaning but clueless dev from slock.it) to formulate a "smart contract" whose meaning cannot be determined in advance by merely inspecting the code: ie, it will always be possible to write a smart contract whose meaning can only be determined after running the code.

Take a moment to contemplate the full, deep (and horrifying) implications of all this.

Some of the greatest mathematicians and computer scientists of the 20th century already discovered and definitively proved (much to the consternation most of their less-sophisticated (naïve) colleagues - who nevertheless eventually were forced to come around and begrudgingly agree with them) that:

Given a "smart contract" written in a Turing-complete language, it is impossible to determine the semantics / behavior of that "smart contract" in advance, by mere inspection - either by a human, or even by a machine such as a theorem prover or formal reasoning tool (because such tools unfortunately only work on more-restricted languages, not on Turing-complete languages - for info on such more-restricted languages, see further below on "constructivism" and "intuitionistic logic").

The horrifying conclusion is that:

the only way to determine the semantics / behavior of a "smart contract" is "after-the-fact" - ie, by actually running it on some machine (eg, the notorious EVM) - and waiting to see what happens (eg, waiting for a hacker to "steal" tens of millions of dollars - simply because he understood the semantics / behavior of the code better than the developers did.

2.3 Properties

The key properties of Orch cyber contracts are:

- Autonomy
- Decentralization
- Auto sufficiency
- Safety
- Precision

Autonomy implies that after a smart contact launches, the deal initiator does not have to participate any more in the process. Smart contracts are not focused on one central server but are distributed by various network points so they can be referred to as being *decentralized*. *Auto-sufficiency* supposes that contracts are able to collect money, realize transactions, distribute resources, issue and spend funds to allow a larger capacity of storage and computation power.

Safety of the data is maintained, since the data in the decentralized registry cannot be lost or modified by the use of cryptography. Precision means that no mistakes can be made due to the absence of hand filled forms.

2.4 Security and risk management

A smart/cyber contract is “a computerized transaction protocol that executes the terms of a contract”. A block chain-based smart contract is visible to all users of said block chain. However, this leads to a situation where bugs, including security holes, are visible to all yet may not be quickly fixed.

Such an attack, difficult to fix quickly, was successfully executed on The DAO in June 2016, draining US$50 million in Ether while developers attempted to come to a solution that would gain consensus. The DAO program had a time delay in place before the hacker could remove the funds, a hard fork of the Ethereum software was done to claw back the funds from the attacker before the time limit expired.

Issues in Ethereum smart contracts in particular include ambiguities and easy-but-insecure constructs in its contract language Solidity, compiler bugs, Ethereum Virtual Machine bugs, attacks on the block chain network, the immutability of bugs and that there is no central source documenting known vulnerabilities, attacks and problematic constructs.

3. Computational Boundedness and Decidable Logic

3.1 Motivation

Preventing loss of computation

Since in Ethereum, the EVM (Ethereum Virtual Machine) is a Turing complete language, so it’s not feasible to static analysis, i.e. we cannot calculate the upper bound on the calculations program will make before its execution, so each program must be provided with gas (which is paid for in Ethereum’s unit of account, ether, to the miner of block containing the transaction) to place the counter and avoid infinite loops. As the program runs out of the gas, the transaction is nullified but the gas is still paid to the miner to ensure they are compensated for the computational efforts they made.

With Turing incomplete programming language like Simplicity, static analysis allows the protocol to place limits on the amount of computation a transaction can have, so that nodes running the protocol are not overly burdened. Furthermore, the static analysis can provide program creators with a general-purpose tool for verifying that the programs they build will always fit within these limits. Additionally, it is easy for the other participants in a contract to check the bounds on smart contract's programs themselves.

Testing of program

They may be buggy, just like any other code. Debugging and testing them is quite involved due to the lack of tools. If Smart Contracts take off in popularity, expect new services and technology focused on doing all types of security checks before they are deployed.

Orch uses Simplicity which has formal semantics. And thus Formal semantics that work with proof-assistant software provide the opportunity for contract developers to reason about their programs to rule out logical errors and to help avoid scenarios like the DAO and Parity's multi-signature program failure. Orch Network employs an interpolation of a programming language Crack as its source language with Simplicity as the target language. The resulting cyber contract programming language is Crackcity.

4. Realtime Unforkable Blockchain Federation and Infinitely-scalable (downward scalable) Monetary System

Orch Network will be deployed on its integral Core Service known as StarCash. StarCash is being engineered as a real-time unforkable blockchain modeled on a hyperfast partition resilient Byzantine Agreement.

Blockchain systems are distributed implementations of a chain of blocks. Each node can issue a cryptographically signed transaction to transfer digital assets to another node or can create a new block of transactions, and append this block to its current view of the chain. Due to the distributed nature of this task, multiple nodes may append distinct blocks at the same index of the chain before learning about the presence of other blocks, hence leading to a forked chain or a tree. For nodes to eventually agree on a unique state of the system, nodes apply a common strategy that selects a unique branch of blocks in this tree. That’s how they avoid forking. There are so many mechanisms to make the blockchain unforkable like the byzantine agreement protocol and many other schemes.

The Unforkable Realtime Blockchain StarCash showcases unparalleled advantages over the blockchains currently in use in cryptocurrency and business settings.

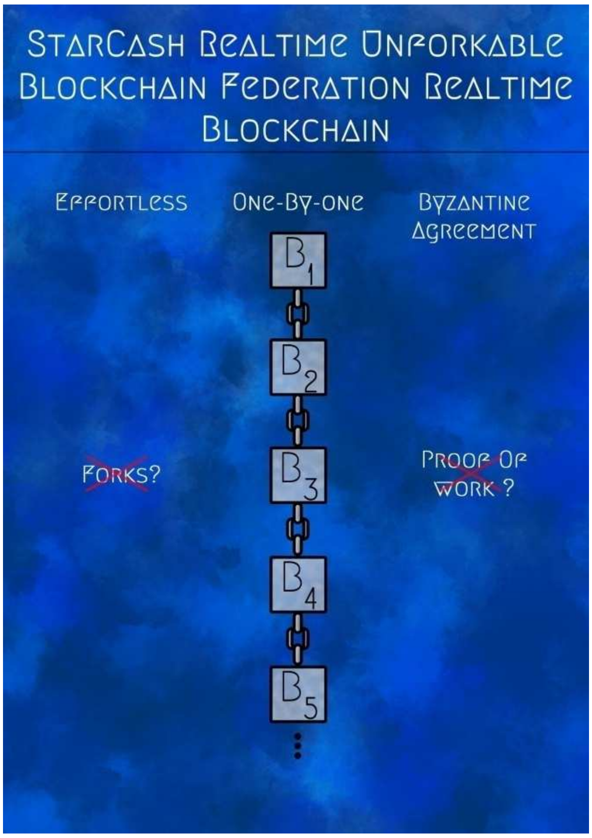
StarCash is based on Effortless One-by-One Partition Resilient Byzantine Agreement that works under > 2/3 honest majority and does not rely on the participants having synchronized clocks.

When honest messages are delivered within a bounded worst-case delay, agreement is reached in expected constant number of steps when the elected leader is malicious, and is reached after two steps when the elected leader is honest. Our protocol is resilient to arbitrary network partitions with unknown length, and recovers fast after the partition is resolved and bounded message delay is restored. When an honest leader proposes a block of transactions, the first voting step happens in parallel with the block propagation. Effectively, after the block propagates, a certificate is generated in just one step of voting.

The design goal of StarCash was building a new method to implement a public ledger that offers the convenience and efficiency of a centralized system run by a trusted and inviolable authority, without the inefficiencies and weaknesses of current decentralized implementations.

For StarCash we use algorithmic randomness to select, based on the ledger constructed so far, a set of verifiers who are in charge of constructing the next block of valid transactions. Naturally, we ensure that such selections are provably immune from manipulations and unpredictable until the last minute, but also that they ultimately are universally clear. StarCash's approach is quite democratic, in the sense that neither in principle nor de-facto it creates different classes of users (as “miners” and “ordinary users” in Bitcoin). In StarCash “all power resides with the set of all users”.

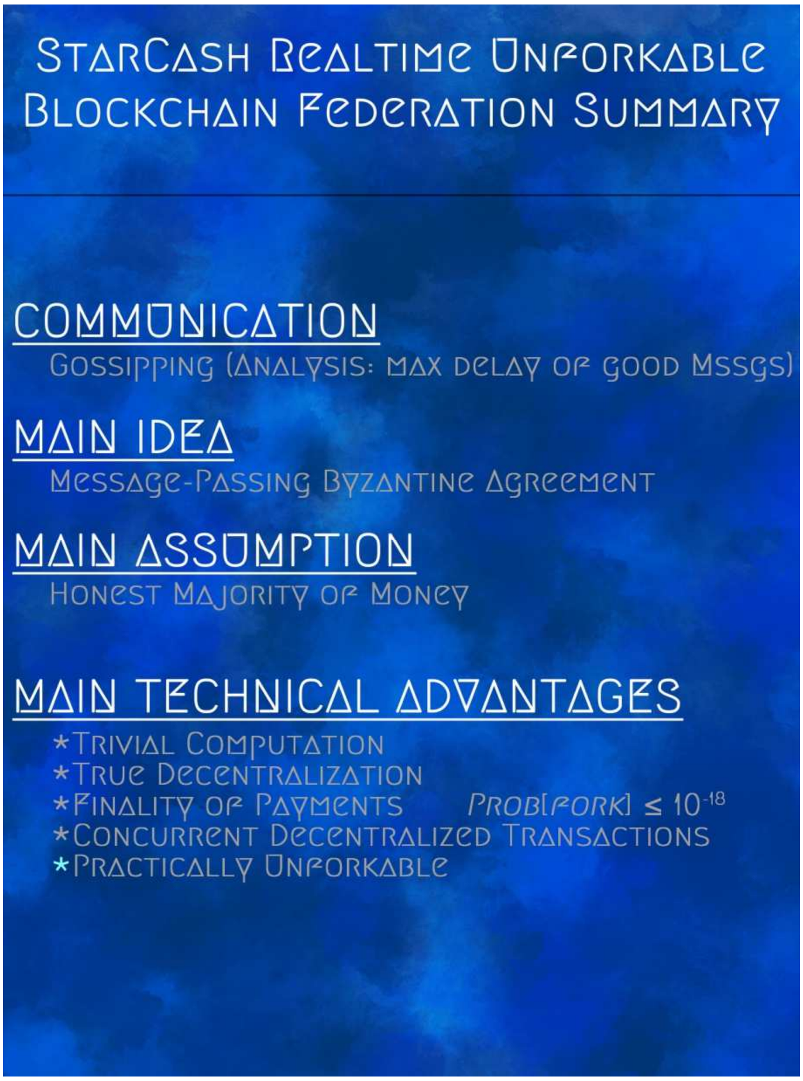
One notable property of StarCash is that its transaction history may fork only with very small probability (e.g., one in a trillion, that is, or even 10^-18).every underlying unconventional mechanism of DexOS based on a single linear Blockchain makes sure there are no multiple blockchains and that's why forking is not possible in the absence of PoW(proof of work) miners.



Here users verify each other's' transactions and there are no special class of miners. StarCash once fully deployed with minimum 10,000 plus nodes will be capable of processing 20 million transactions per second (tps) in parallel. In the near future, a full-fledged StarCash Blockchain Federation running millions of terrestrial and space-based nodes will boost the performance of Orch Network significantly to billions of transactions per second (tps).

These speeds are unparalleled, leaving the closest competitors far behind. In comparison, Visa's network has a peak capacity of around 56,000 transactions per second and the Bitcoin network is limited to around seven transactions per second and Ethereum can process up to 20 transactions per second. Even with Casper fork, Ethereum's performance will not peak beyond hundreds of transactions per second.

StarCash also differs from existing blockchains as it is designed never to fork. Transactions are verified before they are recorded on the blockchain. Orch Network's StarCash Blockchain is practically unforkable. Mainstream traditional blockchains do not order all blocks of transactions with each other. Instead, they tolerate that multiple blocks be appended at the same index of the blockchain, hence causing what are known as forks. To prevent an attacker exploiting these forks to double spend in two of these blocks, these blockchains try to trim out these forks. Instead of curing of the forks, StarCash prevents them: the consensus among the participants occurs before a new block is added so that the blockchain remains a chain.



StarCash Realtime Unforkable Blockchain Federation Summary

COMMUNICATION

Gossipping (Analysis: max delay of good Mssgs)

MAIN IDEA

Message-Passing Byzantine Agreement

MAIN ASSUMPTION

Honest Majority of Money

MAIN TECHNICAL ADVANTAGES

*Trivial Computation
*True Decentralization
*Finality of Payments   *Prob[Fork]* $\leq 10^{-18}$
*Concurrent Decentralized Transactions
*Practically Unforkable

StarCash Blockchain Federation comprising of hundreds of interactive realtime unforkable blockchains is capable of processing billions of transactions per second coming from a potentially unbounded number of clients. It offers a performance that scales horizontally as well as downward, which ensures the security of transactions.

Compared to traditional blockchains which becomes slower and slower as network becomes crowded with more users and transactions, StarCash becomes faster and faster as more users join the network.

Key Features of StarCash

At the heart of StarCash is a hyperfast permissionless Byzantine agreement protocol with leader election, which is safe even in asynchronous networks.

In traditional Byzantine agreements, users try to agree on one of their starting values. In a Byzantine agreement with leader election, users try to agree on a value proposed by a leader.

Our protocol is simple and reaches agreement quickly when the network is not partitioned. In particular, it achieves the following desirable properties under > 2/3 honest majority:

- Fast Agreement. When the network has bounded delay —that is, all honest messages propagate in the network within a given time bound—, an agreement is reached in a constant expected time. In particular, when the leader is honest, his proposed value is agreed upon after two steps of communication.

- Arbitrary Partition Resilience (i.e., Asynchronous Safety). When the network is partitioned (especially, the Adversary has complete control on message delivery and messages may be delayed arbitrarily long), our protocol ensures safety of the system so that no two honest users will finish the protocol with different outputs.

- Fast Recovery from Network Partition. After the network recovers from a partition and restores bounded delay, an agreement is reached in constant expected time.

5. Use Cases and Decentralized Applications of Orch Network

Orch Network will enable and power number of scalable decentralized applications with hundreds of real world uses. Primarily it can easily replace less-secure Ethereum smart contracts written in Turing Complete Solidity language Turing Incomplete Cyber contracts written in Crackcity.

Privacy-preserving Currency and Confidential Global Payment System: Orch Network's ORC native token/currency will have quantum-safe privacy layer zkSTARK implemented in it making it virtually immune to censorship, cryptanalysis and chain analysis for ferreting out payer and payee information as well as payment data by most powerful supercomputers and even quantum computers/quantum circuits in the foreseeable future.

Customizable Decentralized Governance System: Orch Network's StarCash Blockchain Federation' quantum-safe Lattice-based zero-proof voting mechanism can be leveraged for implemen governance protocols of autonomous decentralized organizations.

Unmanned DEXs: Unmanned Decentralized Cryptoasset Exchanges can be built using Orch platform. These unmanned DEXs will be infinitely scalable by design with built-in atomic swaps of both blockchain-based and blockchainfree cryptocurrencies/cryptoassets and can't be controlled or blackmailed by any centralized authorities.

Large-scale Federated IoT Networks: Orch empowers IoT Node owners and IoT device manufactures to implement and monetize securely share private/public IoT resources available on demand by building Federated IoT Networks without revealing any private and identifiable confidential data. Such Network members can be incentivized for sharing statistical aggregated data in ORCs or Colored Appcoins/tokens. This will also make machine to machine micropayments instantaneously and seamless.

Realtime Gaming: Realtime gaming engines, online casinos and gaming apps can be built on Orch platform that can scale to tens of millions of users without associated cost of sever and costly maintenance of centralized cloud infrastructure.

Automated Betting & Lottery Pools: Orch will enable ordinary folks to build private/public betting and lottery pools for optimal high-probability of winning odds for crypto-betting and for buying lottery tickets to guarantee the highest RoIs.

Decentralized DNS Clusters: One can leverage Orch Network and its underlying unforkable realtime blockchain StarCash for assigning human-readable names to IoT nodes, machines, biometrically identifiable humans, secure smart contracts, platforms and so on.

Tokenized Financial Assets and Derivatives Contracts: Orch can potentially tokenize all global/local remittances, payments, trading and hedging. Orch can also be used to create automated and decentralized financial instruments - such as derivatives, bonds, commodities, debt instruments and securities denominated in both cryptocurrencies and fiat currencies by designing and implementing various permissioned and permissionless protocols.

Automated Hedge Funds: Orch with its built-in self-learning and self-planning Degents is the perfect engines for powering Autonomous Goal-driven and Self-managing Crypto-Hedge Funds with portfolios of fiat assets and cryptoassets under their management. These automated funds will not require constant supervision of human fund managers to run investment operations profitably in a sustainable way. The preferred risk-reward profiles and choices made by investors/LPs will be followed and executed strictly without any scope for discretionary violations.

Crypto darkpools: Autonomous Crypto-darkpools for providing privileged peer to peer trading facility and emergency liquidity support to high-frequency/low-frequency wholesale cryptoasset investors/traders.

Temporal Insurance Products: Insurance companies will be able to provide temporal liability insurance using permissioned version of colored ORC tokens of Orch Network. For instance, by employing Orch Network, an insurance company could charge rates differently based on where and under what conditions customers are operating their vehicles. A car driven on a clear day (ascertained by gathering information about the weather conditions from a weather service), in an area where all the roads are repaired (verified with information about road repairs supplied by the Department of Motor Vehicles, for instance), would be charged a lower rate compared with a car that's being operated in bad weather, perhaps on pothole-filled roads.

Global Supply chain: Orch will also be highly useful in the supply chain. One can execute contracts that say, If a consumer receives cash on delivery at this location in a developing, emerging market, then this other product, many, many links up the supply chain, will trigger a supplier creating a new item since the existing item was just delivered in that developing market.

Orch can be leveraged by logistic and shipping companies to power crewless unmanned cargo ships and coordinate global flow of goods and materials.

Realtime Video Communication capable Anonymous Web Infrastructure: Orch enables realtime anonymous protocol for encrypted video calling, video chat and videoconferencing as well as for encrypted voice and texts.

High-velocity Non-sovereign Reserve Asset: ORC, the native currency of Orch.Network will be attractive to long-term institutional and high net-worth investors seeking an alternative high-velocity non-sovereign reserve asset that’s unphysical and highly movable while having the positive characteristics of gold such as the one without any counterparty risks but having none of the downsides of it e.g. gold is difficult to transport and store and so on.

Near-Perfect Coin Mixers: Cryptocurrencies such as BTC, ETH, ZCash, ZCoin, or Monero can be added to DexOS of Orch Network to enable the automatic trading/swaps of any currency for a private currency like Zcash/Zcoin/Monero and back again into the original currency. Since the decentralized exchange does not require any third party to be trusted with users’ data, and since the atomic swap involves no counterparty risk, the result is a nearly perfectly private mixing service.

Decentralized Marketplace App: A marketplace app would typically require the following services: (a) customer reputation and info, (b) payment processing, (c) image storage, (d) item listings. A microservices architecture is advisable for the reasons given above, gaining the advantages of utilizing realtime blockchain federation StarCash.

Transparent Robust StableCoins: An ORC-pegged stablecoin may maintain its peg by exploiting the fact that trade records on a decentralized exchange are on-chain without much latencies and risk of sudden forks. As such, a provably truthful dataset is available for determining whether to mint or burn coins (or freeze and unfreeze them) in order to maintain a peg.

Decentralized P2P Storage

Orch Network can enable decentralized data storage solutions for secure and scalable multicomputing applications including functionally encrypted decentralized and clustered databases on which one can run queries without decrypting.

Permissionless ICO-FCO Platform

Anyone may offer primary and secondary token sales over a decentralized exchange powered by Orch, with no permission required.

Decentralized CDN

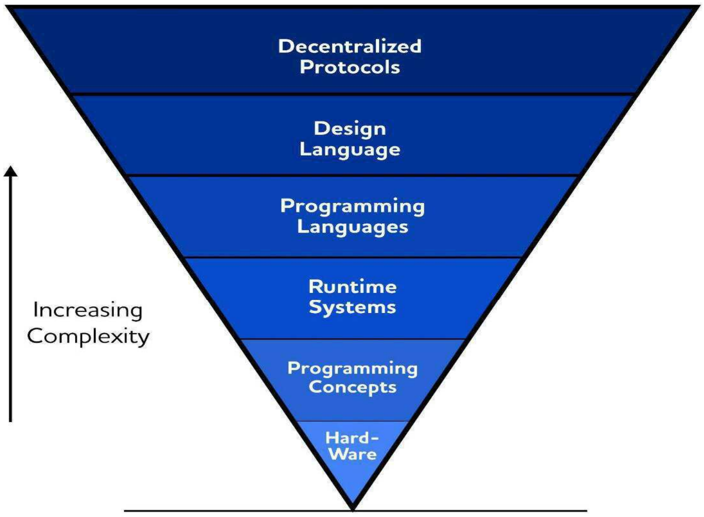
Tens of thousands of P2P Orch Network nodes can serve as points of presence (POPs) for web content caching and delivery through local ISP networks and satnet networks worldwide. Such CDNs can open the gates for censor-free live streaming of videos, voice and data as well as storage of such content safely.

6. Runtimer: A Decentralized Global Code Repository, Compiler and Blockchain Runtime System, all rolled into one

Runtimer is one of the 7 core infrastructure services of Orch Network. Runtimer enables decentralized version control and large-scale peer-to-peer decentralized software engineering supporting both agent-oriented and object-programming in all supported languages including Crackcity, Crack, Python, Rust, C++, Clojure, Haskell and so on.

Software engineers/code contributors can safely and securely interact with software consumers and enterprise users (collectively known as clients) thru their personal mobile agents capable of moving to one or many nodes of Orch Nework along with their execution states and data. All such mobile agents are structured and dynamically configured/reconfigured according to global statefulness(==almost equal to local statefulness with some latencies and partitions) of StarCash realtime blockchain.

With Runtimer, SRS or software requirements specification is no longer sole responsibility of the software engineers rather they deliver adaptive SRSs that can be tuned to the standardized reusable and interchangeable components with various degrees of freedom.

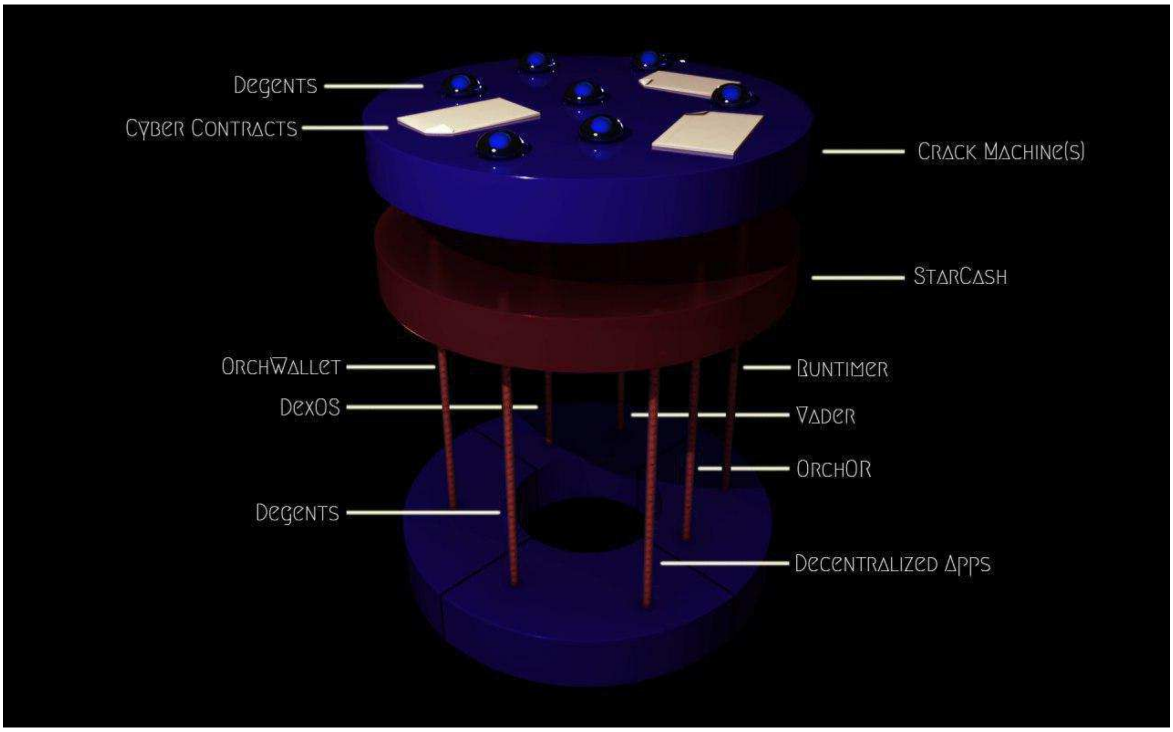


Layered Programming Paradigms

Runtimer incorporates a software assembly protocol that automatically manages and coordinates between various software engineers/code contributors and the client. The result is a dispute-free timely delivery of large-scale software at a fraction of what costs nowadays and even competitive to the quotes of major IT solution providers worldwide.

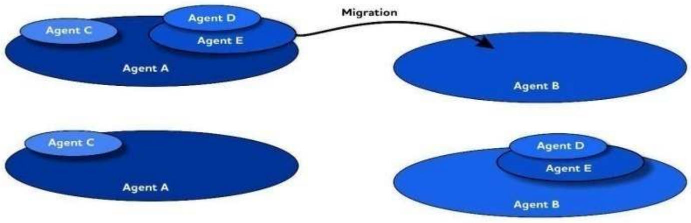
Similarly payments are and escrowed and distributed cryptographically via zero-knowledge proof layer of Orch Network. All payments can be made in ORC coins and/or other major cryptocurrencies of the day e.g. BTC, ETH and XMR etc.

Moreover unauthorized reuse of proprietary code of clients will be minimized significantly.



Degents as a core service as well as an agent and object-based decentralized software engineering platform

In 2018, worldwide spending on enterprise software is expected to reach 391 billion U.S. dollars. That same year, expenditure in the global information technology market is forecast at 3.68 trillion U.S. dollars, while IT services the second largest segment behind communications services presently dominated by MNC and Indian companies, is set to reach 1 trillion U.S. dollars.



Mobile Agents: Migration of hierarchical Degent strong mobile agents between P2P nodes

OrchOr and Degents of Orch Network will have an epoch-making impact on IT services, global enterprise software and information technology markets worldwide.

7. DexOS and Vader: Decentralized censor-proof value exchange and seamless value transfers

DexOS is the decentralized censor-proof operating system and one of core services of Orch Network for instantiating decentralized exchange of values, cryptographic tokens/currencies of Orch Network as well as 3rd-party blockchains/DAG-based networks/Lattice-based networks and other decentralized protocols.

And Vader is the protocol for implementing drivechains that will enable other blockchains and decentralized protocols tap into the advanced and higher-ordered capabilities as well as performance of Orch Network without sacrificing their autonomy. Any 3rd-party decentralized protocol would be able of make their operations infinitely scalable and zero-knowledge proof.

Vader also enables custom child blockchains of StarCash and glue them all together as a Unitary and Integral Realtime Unforkable Blockchain Federation (via orchestrated synchronizations) for boosting performance and flexibility by many orders of magnitude.

8. Crack Machines and Crackcity

Crack Machines are Blockchain virtual machines of StarCash realtime unforkable blockchain that support hierarchical strong mobile agent migration as well as execution of both Turing Incomplete Crackcity cyber contracts as well as zero-knowledge proof decentralized applications.

Crackcity is a Turing-incomplete programming language for programming secure cyber contracts on Orch network. It has been derived by fusing Crack, a general purpose hyperfast script-based language as the source language that compiles directly into native code and Simplicity, the brand new Turing Incomplete programming language for smart contract released by Blockstream in 2017.

9. OrchOr

OrhOr is one of the core services Orch Network. It's a quantum-safe scalable zero-knowledge multicomputation protocol comprising of following subsystems:

- A quantum-safe Lattice-based zero-proof voting system for ORC-token holders and ORC-based or Orchized (thru drivechain implementations of 3rd-party decentralized protocols) tokens/crypotocurrencies.
- A Protocol for transference of proof of pluggable zero-knowledge proof systems including SNARKs, STARKs, recursive STARKs and FHE/SHE-based systems enabling functional encryption.
- A protocol for supporting secure Degents migration and zero-knowledge interaction carrying functionally encrypted data and execution states hierarchically.

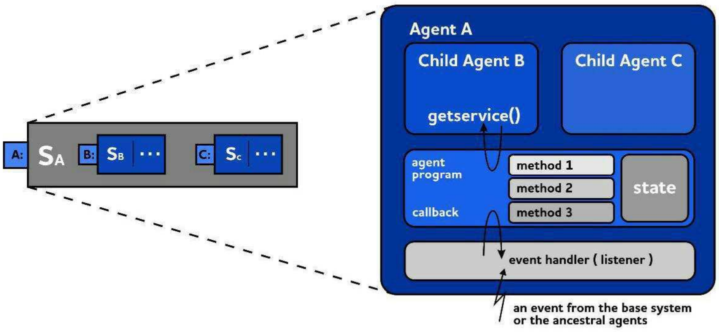
- A scalable quantum-safe zero-knowledge secure multicomputation layer-ensuring guaranteed privacy of transactions, computations, confidential communication and to keep payer-payee data private. It enables an quantum-safe advanced technology of functional encryption known as scalable verification of computational integrity over confidential datasets. So you can basically run queries on encrypted and confidential datasets and still get mathematically verifiable answers without the need for decrypting those in the first place.

  This is the ultimate layer for accessing information strictly on a need-to-know and authorized -to-know basis. Orch Network will have Lattice-based zero-knowledge and Recursive zkSTARK implemented in global data-structures and datasets of its realtime unforkable blockchain StarCash thereby making all orchized (Orch-enabled) applications such as secure multi-computation (SMC) and distributed IoT applications privacy-preserving, functionally encrypted, hack-proof, fully decentralized and quantum-safe by default without sacrificing on speed and scalability.

10. Degents

Degents or Decision Agents are strong autonomous hierarchical strong mobile agents acting on behalf of a blockchain/decentralized service to find and map real-world events to facts/conditions/truth values of Orch cyber contracts. Degents incorporate several machine-learning and pattern recognition models endowing each Degent with higher-ordered self-learning and self-planning capabilities including data smashing and un-supervised distributed learning and planning. Nonclonable Degents can hop from one P2P node of Orch Network to another with its execution state and data either hierarchically carrying other child agents within or alone.

Besides Degents can be used to encapsulate and package pre-tested as well as post-tested finalized software components/systems in the form of live code/agents having their life-spaces (presence by way of their nonclonable execution states) on Orch Network to clients in order to protect un-authorized reuse and execution.



Degents embodied as hierarchical strong mobile agents

Degents will revolutionize the way software is produced, maintained and re-used as well as evolved by forcing industrial-scale automation of software design, creation, testing and maintenance as well as distribution in decentralized way. Besides Degents can serve as engines for powering autonomous systems including those with integral sensors-actors such as drones, UCAVs, androids and robotic mules.

11. Integration with other systems

One system/crypto currencies can integrate with other system/crypto currencies especially those based on PoW/PoS blockchains via use of sidechains and drivechains e.g. Bitcoin and Ethereum. Basically sidechains provide a mechanism to transfer one system's currency to another system's currency securely. Also, a sidechain is a two way peg mechanism. That is an account holder in one blockchain system say Bitcoin can send Bitcoins to another blockchain system say Ethereum and the mechanism of a sidechain will make sure that the Bitcoins from Bitcoin blockchain is converted to ethers based on the real-time conversion rates and then moved into the Ethereum's blockchain system. As sidechains are two way peg system so we also can send Ethers (monetary units of Ethereum) to Bitcoin blockchain. Although sidechains are running projects, there has been found so many flaws and security issues and mining problems.

Due to the problems and securities issues involved in the sidechains, Orch has decided to use Hashrate Escrows and Drivechains pioneered by Paul Sztorc in order to move value from Bitcoin and other forks of Bitcoin e.g. Bitcoin Cash and Bitcoin Gold to Orch.We plan to merge a drivechain into Bitcoin Core, then we will insert a few parts of our sidechain template and insert a few parts into StarCash — the code for deposits (crediting BTC Orch for side-to-main transfers) and withdrawals (aggregating side-to-main transfers and broadcasting an easily-visible). Same mechanism will be repeated for other forks of Bitcoin such as Bitcoin Cash and Bitcoin Gold and numerous other blockchains and decentralized protocols.

This will help us move almost a quarter trillion dollar and growing value from BTC, BCH and BTC Gold to Orch Network. Vader as one of the seven core services of Orch Network fits these roles of integrator and value exchanger/shifter.

12. Future Work

- Orch Network uses Simplicity as its target language which has no unbounded loops or recursion. It is possible to build smart contracts with state carried through loops using covenants without requiring unbounded loops within Simplicity itself. Bounded loops needed by our Blacke2b implementation, can be achieved by unrolling the loop. Because of sub-expression sharing, this doesn't unreasonably impact program size. We do not directly write Simplicity, rather we use functions written in Coq or Haskell to generate Simplicity. These languages do support recursion and we use loops in these meta-languages to generate unrolled loops in Crackcity.

- Simplicity has no function types and therefore no higher-order functions. While it is possible to compute upper bounds of computation resources of expressions in the presence of function types, it likely that those bounds would be so far above their actual needs that such analysis would not be useful. But we plan to add higher-order functions to Crackcity.

- **Incentives:** In order to encourage Orch users to participate, i.e., be online when selected and pay the network cost of operating Orch, the system need to include incentives in form of a reward mechanism by minting ORCs(Orch Network tokens and monetary units) out of the hardcap supply as per a controlled Emission rate. Designing and analyzing an incentive mechanism includes many challenges, such as ensuring that users do not have perverse incentives (e.g., to withhold votes), and that malicious users cannot “game the system” to obtain more rewards than users who follow the protocol.

- The design and implementation of Crackcity, a Turing Incomplete interpolation of Crack language is undergoing.

- Building a Orch Network testnet

- Developing a Lightwight standalone Orch Wallet

- Deploy Space-based infrastructure by launching an Alternative Space Backbone ASB Freedom as per our Roadmap and Self-governance protocol.

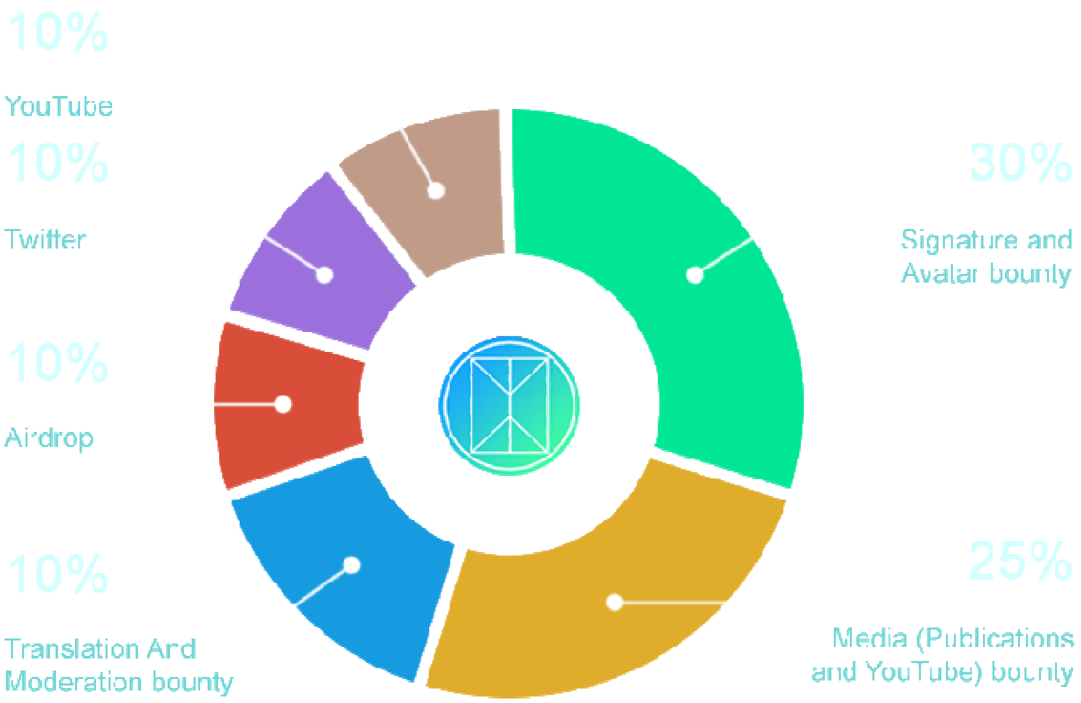
13. Token Units and Policies

13.1 Token Supply & Monetary Policy

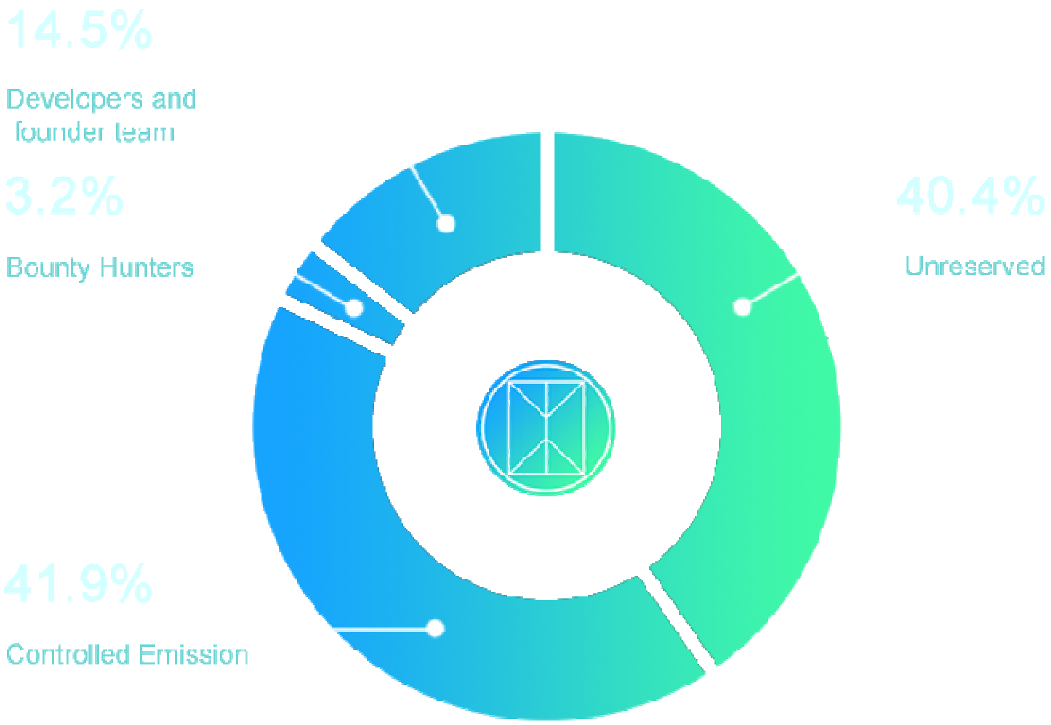
Total coins planned to be launched: 62,000,000,001
Maximum Supply (Hardcap): 62,000,000,001
Reserved for developers and founder team: 9,000,000,000
Reserved for Bounty Hunters: 2,000,000,000
Reserved for Controlled Emission (minting) to reward peer-to-peer random users-payment validators: 26,000,000,001
Divisibility of token units: Infinite (downward infinitely-scalable)

- Limited Supply, with total set by the token sale.
- Genesis allocation to fund creation, development, deployment, and reward creators.
- Backed by useful services and applications: Demand for Orch-powered applications e.g. privacy-preserving cryptocurrency and confidential hyperfast global payment system, decentralized software engineering infrastructure, scalable gaming infrastructure, unmanned

DEXs, large-scale Federated IoT Networks, tokenized financial assets and derivatives contracts, crypto darkpools and automated hedge funds



Bounty Allocation



Orch Coin Allocation

14. References

1) First PoW protocol Hashcash:
http://hashcash.org
https://raiblocks.net/media/RaiBlocks_Whitepaper__English.pdf

2) Tauchain Decidable Generalized Blockchain:
http://tauchain.org/tauchain.pdf
https://bitcointalk.org/index.php?topic=950309.0t

3) Hash Rate Escrows- Drivechains:
http://www.drivechain.info

4) zkSTARK Technical Paper:
https://eprint.iacr.org/2018/046https://eprint.iacr.org/2018/046

5) Simplicity Turing Incomplete Language:
https://blockstream.com/simplicity.pdf

6) Algorand:
https://people.csail.mit.edu/nickolai/papers/gilad-algorand-eprint.pdf

7) Plasma: Scalable Autonomous Smart Contracts: Joseph Poon of Lightning Network and Vitalik Buterin of Ethereum Foundation:
https://plasma.io/plasma.pdf

8) Absolute Undecidability:
http://logic.harvard.edu/koellner/QAU_reprint.pdf

9) An Institutional Investor's Take on Cryptoassets: John Pfeffer
https://s3.eu-west-2.amazonaws.com/johnpfeffer/An+Investor%27s+Take+on+Cryptoassets+v6.pdf

10) Undecidability of Generalized Collatz Problem:
http://people.cs.uchicago.edu/~simon/RES/collatz.pdf

11) Theoretical Foundations for cryptography: 2
https://www.scottaaronson.com/papers/pnp.pdf

12) Crack Languange Guide:
http://crack-lang.org/manuals/Manual-1.3.html#the_crack_programming_language_guide