WHITEPAPER

Version 0.5

LALIN Hugo

contact@virgocoin.io

VIRGO : a high-performance cryptocurrency for the creation of decentralized applications.

Table of contents

INTRODUCTION

Since its emergence, blockchain has demonstrated that it can allow the creation of a more fluid economy, with no trusted third party.
This makes it possible to improve many industries where the need for trust is a problem, such as finance, data management, or even supply chains.
However, this technology, which should already be a standard in these sectors, is struggling to take hold due to its complexity.
Indeed, there are currently as many different blockchains as there are projects, thus mitigating its advantages.
While blockchain enables more fluid exchanges, we find ourselves constantly having to own and exchange a large number of currencies depending on our use, with real complexity and difficulty.
At the same time, we have to trust in a multitude of different blockchain technologies and actors who make up this multitude of projects, thus reinforcing contradictions.
This multiplication of the number of solutions is in part due to fundamental problems related to what is called the "blockchain trilemma", namely: scalability, security, and decentralization.

All of this together means that many projects with innovative solutions are struggling to get adopted by a wider public, which, being most of the time poorly educated concerning the crypto world, finds itself lost behind a wall of incomprehension.

Much more than a cryptocurrency, Virgo's main concern is to provide developers with the best possible tools for creating and distributing decentralized applications.
These tools mainly consist of a high-performance cryptocurrency responding to the blockchain trilemma, then, in a second step, of a dApps distribution platform in the AppStore style, easy to use and fully decentralized.

VirgoLedger uses a DAG (Directed Acyclic Graph) for data structure and proof of work for it’s consensus method, introducing for this a new type of transactions forming a chain above the DAG, permitting to give transactions a final order.
By combining DAG and “classic blockchain”, Virgo manages to bring together the best of both worlds to offer a consensus method that is at the same time scalable, secure, and decentralized.

Apps became an important part of our digital ecosystem.
With their increasing use, key issues such as availability, throughput, assurance of software authenticity, protection of intellectual property and rights of application developers have become issues that have a significant impact on this ecosystem.
In our whitepaper, we present the first decentralized application distribution platform that uses a distributed file system to address the aforementioned issue.

This “store” of applications operates autonomously. Easy to use, it can remedy some major inefficiencies observed today.
The technology can eliminate all intermediaries and will create a new ecosystem:
Users will get virgocoins (VGO) which, once stored in their wallet, will be usable directly to buy services and use applications.
Some applications may be free to use or even rewarding thanks to the distribution of VGO via integrated ads or by passive mining (the user's PC will be mined while it is not in use).
With our model, once users acquire virgocoins, they can simply spend them on their favorite apps, creating a circular economy.

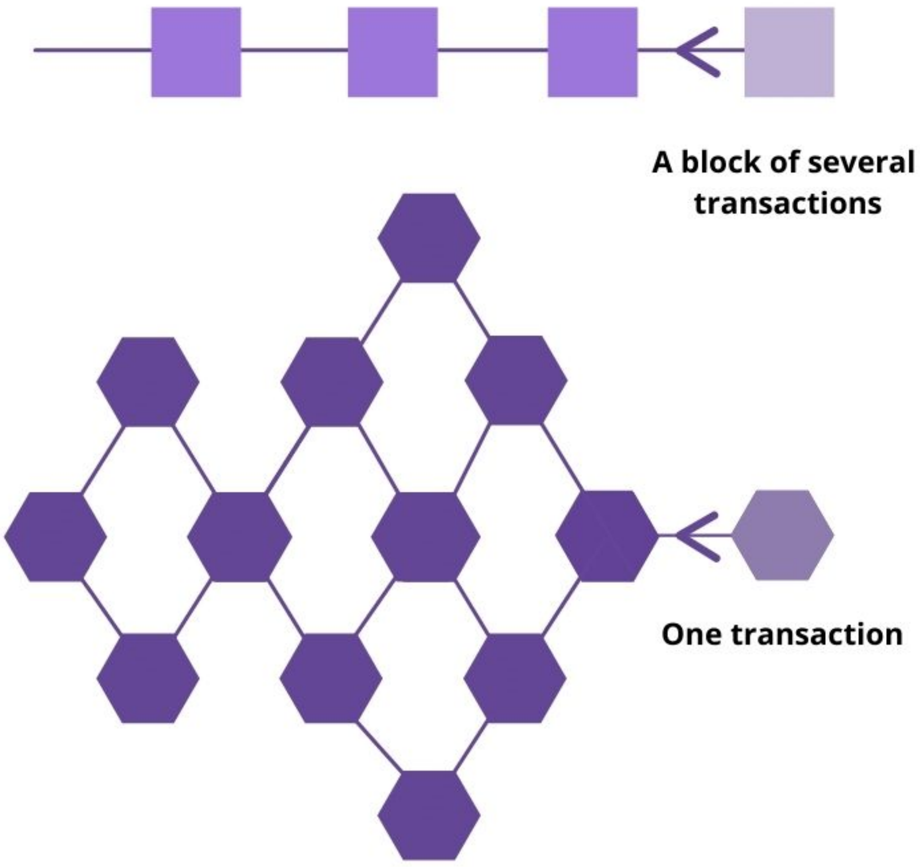
It should also be noted that the approval process for existing applications is complex today, managed by centralized structures with complex quality assurance flows and non-transparent distribution policies. By using Virgo, app approval will be made universal and more transparent.
To guarantee the decentralization of the application validation process while protecting users, anyone can create and maintain a list of recognized applications and user reviews about it. Popular lists are offered by default to the user, but one can import any list to enlarge it’s possibilities.

I - VIRGO LEDGER

1 / Operation

1.1 / The Directed Acyclic Graph

Unlike most cryptocurrencies, VIRGO is not based on a blockchain, but on a Directed Acyclic Graph (or DAG), which can be thought of as a two-dimensional blockchain.



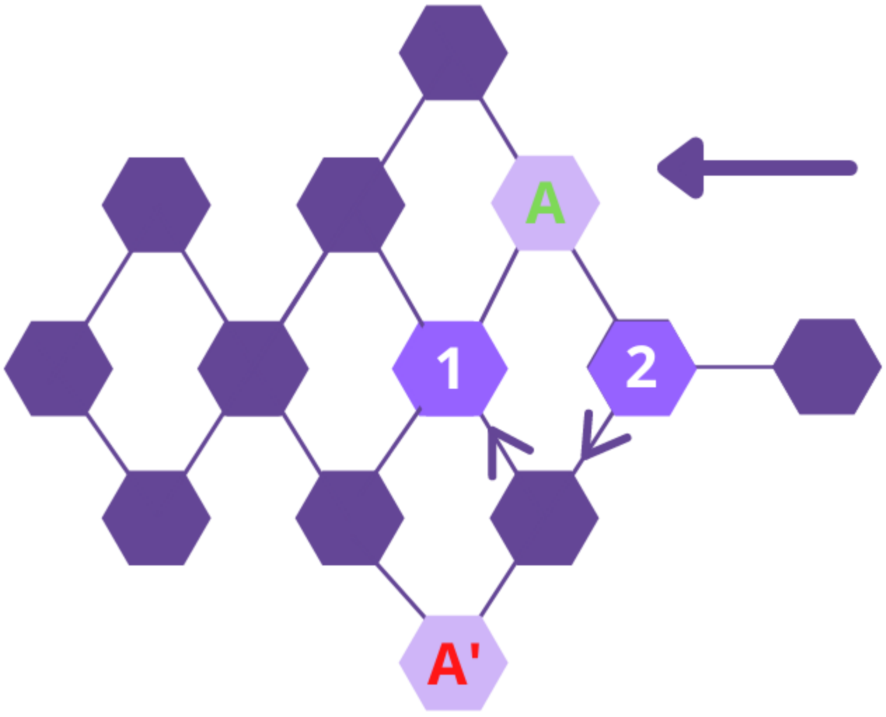
As can be seen in the figure above, the main difference is that each vertex (understand block) can link to multiple “parent” vertices (appeared before) instead of just one on a classic blockchain.
In a classic cryptocurrency like bitcoin, grouping transactions into blocks makes it possible to give an order to them, the blocks being generated at regular time intervals via mining and each referencing the block before it, the whole forming a chain.

We, therefore, know that a transaction is validated before another by looking at the order of their respective blocks, this allows us to count in the right order and prevent someone from spending several times the same money.

The blocks must be spaced out enough and small enough so that the majority of the computers forming the network have time to receive the most recent one, otherwise two parts of the network can receive a different block, causing two different versions of the blockchain, we then speak of “Fork”. This limits the number of transactions per second that can be managed by the network and if this limit is reached the transactions with the highest fees are preferred, making emitting a transaction very costly (Bitcoin has for example already reached over $70)

With VIRGO, the graph allows us to get rid of blocks. Each vertex is now a single transaction and the order of this is done naturally because each transaction refers to one or more previous transactions (this is what forms the graph, or even figure 2), so they no longer need to arrive simultaneously on all computers in the network.

But using a DAG creates a new problem: transactions do not necessarily have an explicit order.



For example, in this figure, we can start from transaction 2 to go to 1 by walking back the graph, so we know directly that transaction 1 was issued before. But there is no path between A and A ' walking back only, so we cannot know which transaction was issued first

If A and A' do not try to spend the same funds, this is not a problem, but if this is the case then it is necessary to decide an order for them. This cannot be decided at random because we would get a “fork”, nor can we simply refuse the two transactions otherwise anyone could cancel a payment and get their money back by issuing a new transaction competing with the old one.

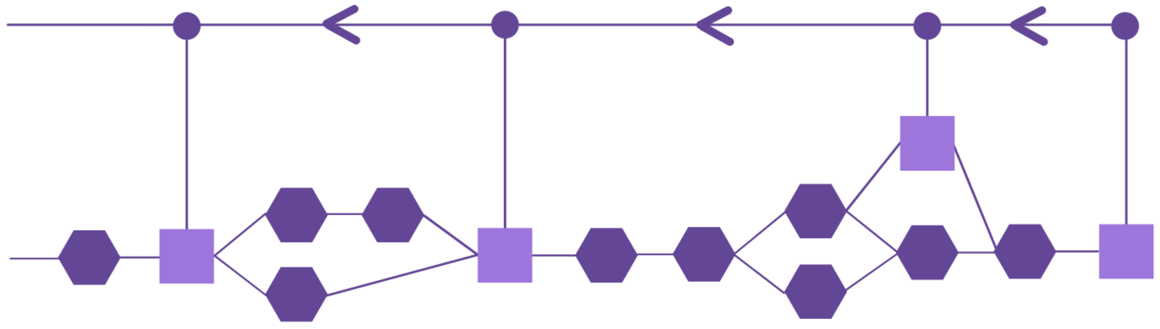
1.2 / The beacon chain

To solve this problem and date transactions, we are introducing a new type of transaction: beacons.
A beacon transaction integrates with the DAG by referring to one or more parent transactions, like a normal transaction; But in addition, it is referring to the previous beacon transaction.
In this way, we obtain a chain of beacon transactions, similar to a classic blockchain, covering the entire DAG.

To preserve Virgo's decentralization, anyone can issue a beacon; But it is necessary to preserve a form of chain for the rest of the solution; We are therefore going to implement a proof of work system similar to Bitcoin's one:
For a beacon transaction to be considered valid, the value of its hash must be less than a certain value.

The minimum required value is inversely proportional to the difficulty, a value adjusted by an algorithm taking into account the time between the last beacons, so that the spacing between each of them is constant even if the computing power assigned to the task varies (speed of equipment, number of participants).

Someone trying to create a new beacon has to increment an arbitrary value, called a nonce, to that beacon until its hash value follows the requirement, which takes time relative to its computing power compared to the rest of the network.



Several beacons referring to the same parent may be found in a short time, especially in the case of network overload.
In this case, the beacons considered valid are those forming a chain that represents the most proof of work (therefore the most accumulated difficulty).

Each valid beacon confirms the transactions between it and the valid beacon it is a child of.

To do this, we will start from the beacon transaction being confirmed and walk down the graph from transaction to transaction until meeting an already validated transaction; We apply the following confirmation logic to each transaction:

1. If the transaction has already been processed by another beacon stop here.

2. If the transaction tries to spend an input that has already been spent or that has been refused, it is refused and continues with its parents.

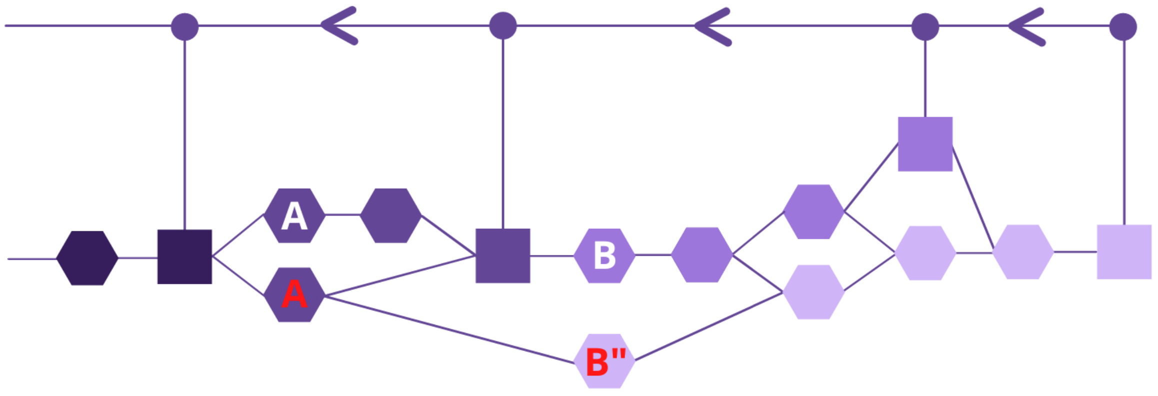
3. We check if another pending transaction tries to spend the same funds, if so, add the transaction to the list of conflicting transactions and continue with its parents.

4. If the transaction did not trigger the previous verifications, confirm it and continue with its parents.

Once all the transactions have been processed, perform the following logic on each conflicting transaction:

1. If the transaction conflicts with one or more transaction being confirmed by this beacon, refuse it (so in the end the competitors will also be refused)

2. Otherwise confirm it, since the concurrent transaction will only be processed by a subsequent beacon (and will therefore be refused).



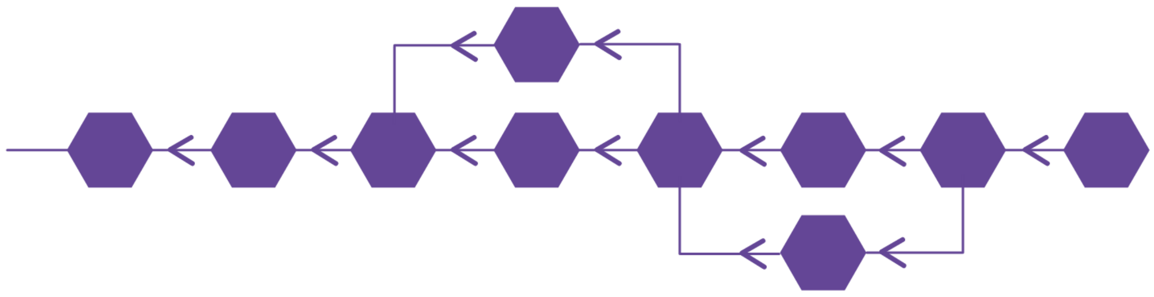
For example here, **A** and **A "** are concurrent, both are refused because they are confirmed by the same beacon.
B and **B "** are also concurrent, but here **B** was treated a beacon before **B"**, so **B** is confirmed while **B "** is refused.

It should be noted that a conflict between two transactions having an explicit order cannot exist, because the child transaction would automatically be deemed invalid.

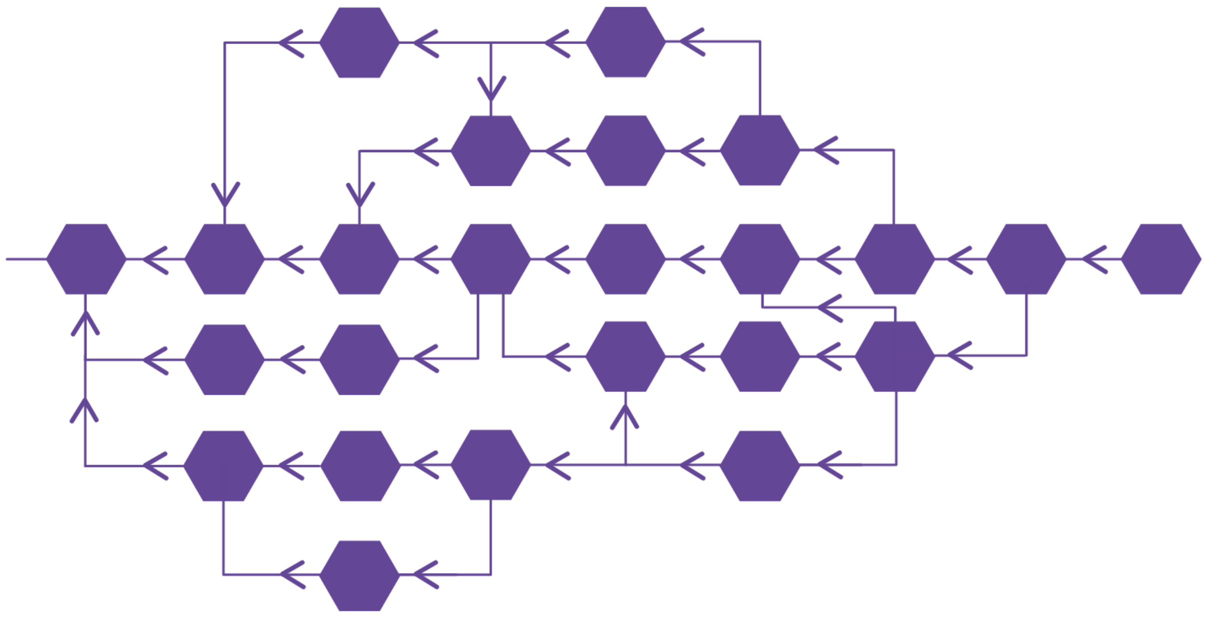
The number of transactions that can be between each beacon is not limited.
Thus, unlike a blockchain, the number of transactions per second is not limited by the size of the blocks and their transmission speed;
But rather by the speed at which the nodes forming the network can process these, and therefore the computing power of these and the optimization of the Virgo code.

1.3 / Network overload

Nodes are those that tell users which transactions to choose as a parent. To grow the DAG, these provide as parents the childless transactions at the time of the request. So if the number of transactions per second is low, few users will choose the same parents for their transactions, so the DAG will look like a chain of transactions.



Counterwise, when the network load is high, several users will attempt to create a transaction at the same time, and will therefore choose the same parent.
The graph then becomes larger, with many transactions at the same height.



This principle also applies logically to beacon transactions:
If the network is overloaded, the number of beacon transactions evicted will increase because miners will take longer to see the new beacon transactions.

Each evicted beacon is one less opportunity to expand the main chain and therefore to secure the network, even more, it is wasted computing power.

In addition to this, the smaller a miner, the more likely he is to have his beacon evicted when he finds it, which increases the centralization of mining:
If A and B find a block at the same time, and A has 30 % of the computing power while B 10%, A will have a 70% chance of seeing his beacon evicted while B 90%.
This means that if the network is overloaded enough for the chances of producing a stale beacon to be great, A will de-facto be more efficient than B by its size.

To avoid this, we can simply implement the GHOST protocol proposed by Yonatan Sompolinsky and Aviv Zohar in December 2013, already implemented in Ethereum.

The GHOST protocol proposes to take into account the stale beacons in the calculation of the longest chain, but also, as proposed this time in the Ethereum Whitepaper, to attribute a mining reward to these.
Thus, the computational effort put into a stale beacon is not wasted, and miners are rewarded in all cases, reducing inequalities.

Stale beacons must not be accepted in all circumstances: they must have a depth limit to avoid allowing miners to obtain a reward for mining on low-difficulty beacons.

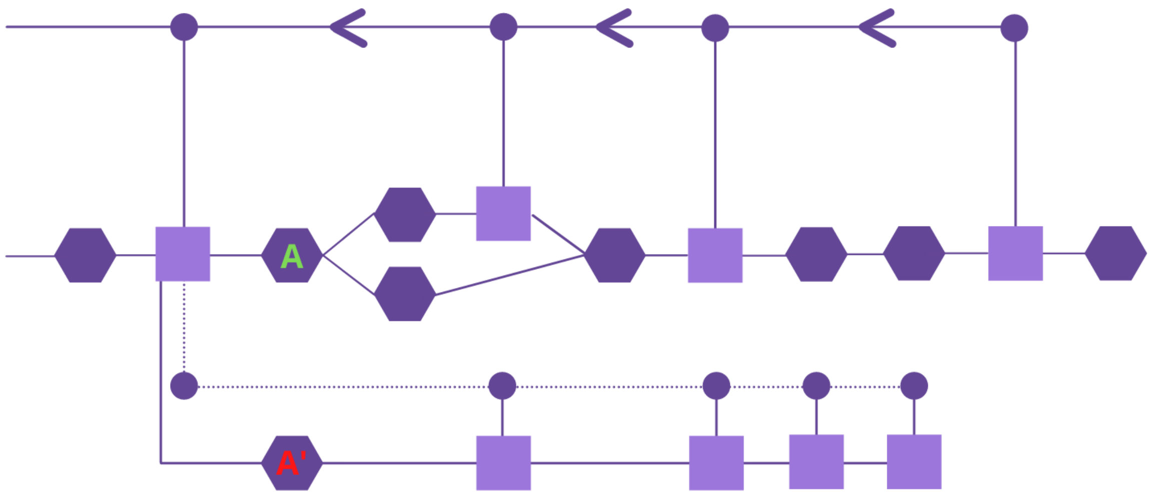
As popularized by Ethereum, we will use the term uncle for accepted stale beacons.

The implementation of GHOST in Virgo, therefore, works as follows:

1. A tag must specify a parent beacon, and optionally one or more stale beacons
2. An uncle beacon included by a beacon B must have the following properties:
3. It must be a direct child of an ancestor of k generations of B, where 2 <= k <= 7.
4. It cannot be an ancestor of B.
5. It cannot be included in a parent of B.
6. For each uncle included, the miner of B gets 12.5% additional reward, and the miner of the uncle in question the remaining 87.5%.

1.4 / Reversing a transaction

As with any distributed ledger, transactions on Virgo can be canceled under very specific circumstances, this results from the capacity of the network to resynchronize itself in the event of a fork.
For an attacker to be able to roll back a transaction, he must create a chain of beacons that is larger than the main one and include in this new chain a transaction competing with a previously accepted transaction.



Here, for example, the attacker first sends a transaction (**A**) to a merchant in exchange for a service. Once the service has been delivered, the attacker creates a transaction **A'** spending the same funds as **A**, but pointing to an address that he controls and not that of the merchant.
A' will first be denied, but the attacker will then create a beacon chain surpassing the one where **A** is present. The illicit chain becoming the main one, **A'** will be accepted and **A** refused.

It should be noted that such an event does not allow the attacker to make arbitrary changes such as creating coins or spending funds that do not belong to him, an invalid transaction remains invalid. So, the only thing he can do is recover the funds that he has already spent.

As the attacker tries to generate its competing chain, the rest of the network continues to grow the main chain. Thus, for his chain to have a chance of becoming the largest, the attacker must have at least more than half the computing power of the network, hence the term 51% attack.

This type of attack is even more restricted in Virgo:
With a classic blockchain, if the main chain is replaced by an empty chain, the miners will have to re-include the transactions it contained, taking a time proportional to the number of canceled blocks.

With Virgo, once the attack is over, honest miners will take transactions from both chains as their parent and therefore these will be re-confirmed in a beacon, i.e. a few seconds.

1.5 / Issuance and transaction fees

The Virgo network has a base currency, Virgocoin, which will be used as the main means of transferring value on it.

Its main denomination will be VGO, which will correspond to 10^8 base units.
In other words, Virgocoin will support precision of 8 decimal places.

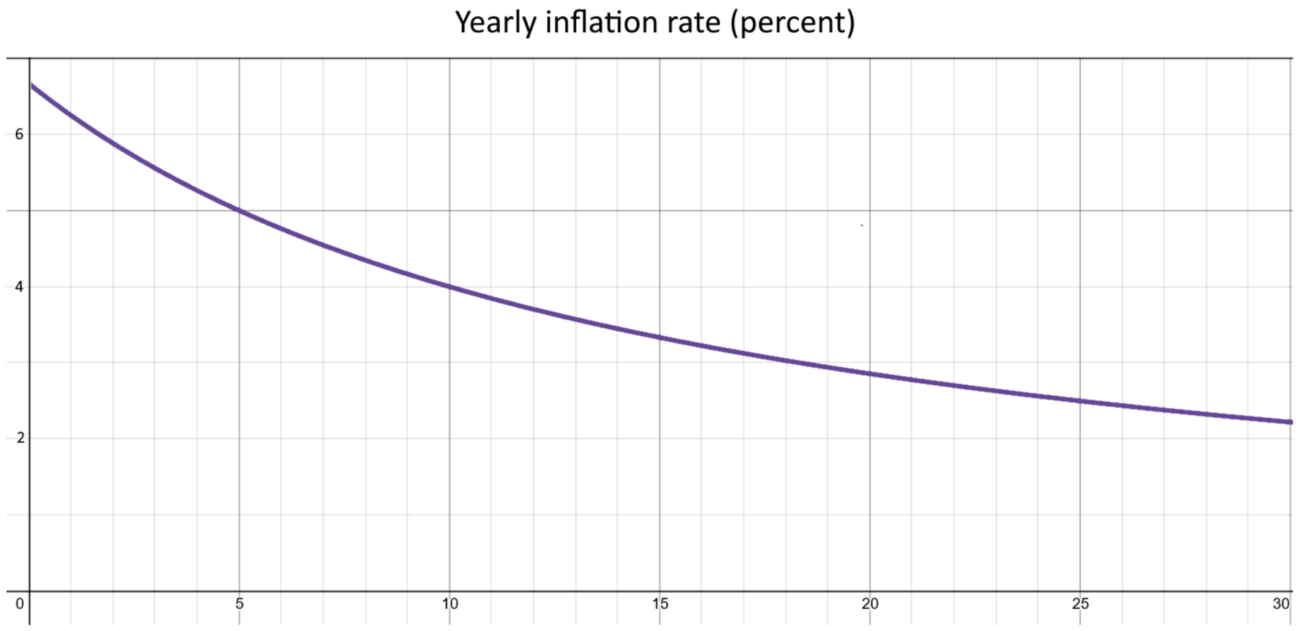
To support development, 30,032,000 VGO will be pre-mined and distributed as follows:

- 15% will be reserved for the team
- 15% will be distributed via development bounties
- 10% will be distributed via Marketing campaigns
- 60% will remain in reserve to be either sold during an ICO or burnt if necessary. This 60% will not be claimable by the team.

Creating a beacon and therefore securing the network is costly in computing resources, and economic motivation is necessary to encourage the members of the network to participate in its security.
Each beacon will therefore have by convention only one output, assigning a defined number of new coins to the issuer of the transaction.
So even if someone happens to account for a large percentage of the network's computing capacity, it is very likely that they have more interest in being honest and racking up the rewards rather than attempting to double spend.

This reward will be set at 2 VGO per beacon, and one beacon will be generated every 30s on average, which will result in the creation of 2,102,400 VGO per year.

Even if, unlike Bitcoin, the reward per block does not tend towards 0, this is still the case for the annual inflation rate:



The choice not to limit the maximum number of coins in circulation was made to keep a sufficient incentive for miners, but also to avoid wealth centralization.
It is also known that a certain amount of coins will be lost each year for various reasons; As annual inflation declines, it and the number of losses per year will possibly reach equilibrium and the number of coins in circulation will then become stable, and even reach slight deflation.

Thanks to the constant reward and the absence of blocks, Virgo could in theory operate without transaction fees.
But removing any fees would open the door to abuses like Penny-Spend attacks, unnecessarily increasing the size of the DAG and the storage space needed to maintain a node.

To limit abuse, a transaction must include fees proportional to the storage space it takes, a price per byte being defined in advance in the protocol.

These transaction fees will not be distributed to the miners but rather burnt, to lower the inflation due to mining.
This will get rid of fee estimation algorithms which are extremely complex and which in the end very often overestimate the necessary transaction fees.

The profitability of mining will also become more constant and predictable, making the security of the network more stable, while keeping a low cost per transaction and therefore suitability for micro-payments.

1.6 / Centralization of mining

Mining for any cryptocurrency based on proof of work function in the same way: Miners calculate the hash of a block (or beacon transaction in our case) with a precise algorithm, for example, Sha-256, repeatedly adding random information until the value of this hash matches the criteria defined in the previous block.

The problem with this algorithm is that it is often possible to create machines specialized in this task (ASICs, for Application-Specific Integrated Circuit), which are orders of magnitude faster than hardware available to the general public, such as CPU.

Mining then quickly becomes an activity that is profitable only at industrial levels and in regions where the cost of electricity is cheaper.
This centralizes the network and thus makes it more sensitive to attacks and regional outages.
For example, as of this writing, China accounts for over 60% of the overall Bitcoin hash rate, and a power outage in one part of the country has cut off 40% of the network's computing capacity, causing fees to explode.

To prevent the centralization of the network, Virgo will use RandomX as its hashing algorithm.

Initially developed for Monero, RandomX is an algorithm optimized for general-purpose processors.
It uses random code execution (hence the name) as well as several RAM techniques to minimize the efficiency advantage of specialized hardware (ASICs and FPGAs).

Thus, the only high-performance hardware when mining via RandomX is the general-purpose processor included in any computer.
This will make Virgo mining accessible to a large number of people, without the need for significant upfront investment.
Also, choosing a fairly uncommon algorithm will protect Virgo from Hashrate-Attack during his childhood.

1.7 / Performance

VIRGO, due to its design, has no specific limits in terms of transactions per second. This means that its performance depends only on the optimization of its code, the computing power of the machine that executes it, and the speed of the internet network, unlike cryptocurrencies based on a blockchain which, for most are limited by the size and speed of block transmission.

The current implementation manages to process over 1000 transactions per second on an AWS EC2 'c3.2xlarge' server without any optimization, which is promising and already more than enough for the young days of Virgo.

When adding a transaction, the majority of the processing time corresponds to the verification of the cryptographic signature of this one. The method currently used is largely optimizable.
It is possible to implement a concept of super-node, where several servers controlled by the same entity work together to improve the processing capacities of the whole (for example a main server, which manages the DAG, and 'slave' servers which deal exclusively with the signatures verification process).

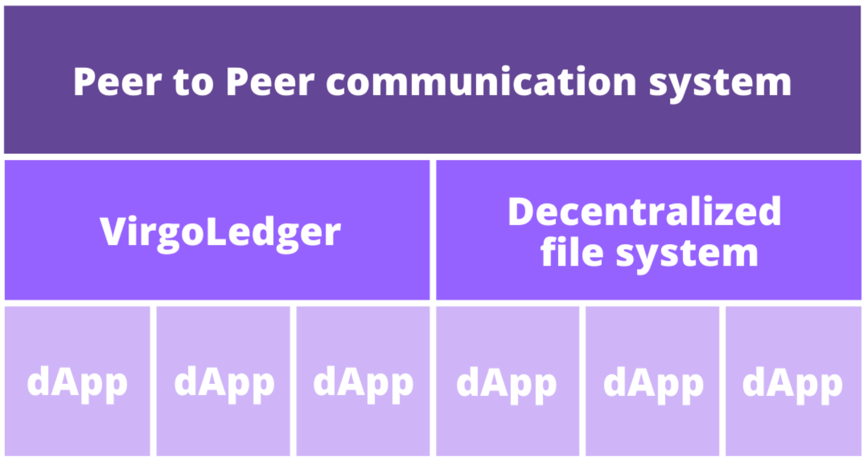
II - Decentralized application store

As said in the introduction, we do not believe that a payment protocol,
as powerful, secure, and decentralized it can be, can by itself lead
to more global adoption of the technology.

The real potential of distributed ledgers is when you combine them with other peer-to-peer technologies, to create applications and services that are more robust and performant than their centralized counterparts while avoiding any form of censorship or authority.

Virgo's primary goal is to create a suite of open-source tools and protocols that allow anyone to create a distributed application as easily as possible.

VirgoLedger is the first brick of this suite, but others, such as our peer-to-peer communication or content distribution library already exist or are to come.
These bricks, which can be dependent on each other, will serve as a common basis for applications, which can themselves also be seen as additional modules to Virgo.



They can be distributed through a decentralized platform, similar to AppStore or Firefox extensions, to be as accessible and easy to use as possible.
The final goal is to offer the user, through a single platform, a complete ecosystem of distributed services.

Whether for cloud storage, a VPN, decentralized finance, communicate securely or simply make payments;
The user will only have to install the desired service in one click via his wallet and use it.
Simplicity for global adoption.

1 / Operation

In essence, VirgoStore will be a Virgo wallet module.
It will allow users to consult a list of applications that are complementary to the wallet, then download and install them with one click.

The applications can then communicate with the wallet, for example, to ask the user to make a payment or to allocate funds to it.

1.1 / Decentralization

To keep Virgo as decentralized and resilient as possible, applications will be downloaded via a peer-to-peer file system, such as BitTorrent or IPFS.

To ensure the integrity of the downloaded files, the wallet will know in advance the hash of the various applications, so it will only have to compare the hash of the downloaded data with the reference hash to prevent any malicious alteration of the apps.

The applications, their information (such as their description or website), the hashes of their versions but also the user's opinions about them will all be listed in databases. These databases can be created and maintained by anyone, and can then be imported into the wallet in one click.

The maintainers of these databases will sign each version of them and then distribute them via the same file system as the applications.
Thus, the wallet will be able to keep up to date with the latest applications and comments, without having to rely on a centralized system.

To simplify installation, the wallet will by default embed the most popular lists, one of which will be maintained by the Virgo team.

In the end, the result is a fully decentralized application distribution system that does not compromise user security or ease of use.