METASAFE

Contributors: Naveen K. Gutti, John Wu, Arup Bhattacharya

Gryphon Online Safety, Inc.

Version 1.2

This white paper reflects the current ongoing MetaSafe project based on Web3 technologies. Since Web3 technologies are still evolving, this paper will be updated periodically, as and when required. Please visit this site to review the latest version of this living document.

Acknowledgement

Much of its content and MetaSafe project is inspired by the many ongoing research, technical papers and other Web3 projects.

1. Abstract

In the current Internet infrastructure (Web 2.0), information is stored and flows through centralized servers [1]. Every time we create an online account, visit a doctor's office, call our insurance company, or just purchase something online, we give away our personal information. Sometimes that information can surface on the dark web where hackers can use it for identity theft or worse. A few major drawbacks of storing our information on centralized servers is increased exposure to hacking as well as the exploitation of our personal data by corporations for profit without regard to our privacy.

The MetaSafe platform is a *decentralized* storage and peer-to-peer network that enables anyone with an Internet connection to store their information securely from any computing device and privately retrieve it on demand or share it at will. Due to its decentralized nature, data is virtually hack-proof and not subject to any commercial exploitation. To achieve this, MetaSafe uses blockchain technology, the same technology that safeguards cryptocurrencies but with some unique enhancements. These enhancements include proof-of-stake, proof-of-bandwidth, and proof-of-storage consensus protocols, a utility token to incentivize network build-out and track value exchange, and finally the ease of running the validation nodes on millions of standard routers. Modern routers with powerful processors are an under-utilized piece of hardware in most homes and the always-on, always-connected nature of these devices make them a perfect system for running MetaSafe.

As a utility blockchain, many security, communications, and privacy applications are expected to be built on MetaSafe. At launch, a password manager (Passport by MetaSafe) will be available that will function like a traditional password manager but use MetaSafe blockchain for secure storage at a fraction of the cost of existing systems. Password and identity management already has wide adoption and is expected to grow to over $3 Billion in the next few years [2]. A major portion of the revenue generated from MetaSafe applications will be shared back out to the MetaSafe validators.

MetaSafe is bringing the next billion users to Web3 by making it simple to be a validator and baking utility into the blockchain to capture the value of existing and future applications.

2. Introduction

Ethereum [3] and other blockchain technologies used to safeguard cryptocurrency demonstrated the advantages of a decentralized network without the need for a central authority. Many of the current

blockchains use proof-or-work to mitigate cybersecurity risks. This has proven cost prohibitive for scaling due to high energy consumption [4], slow transaction speeds, high transaction costs, and lack of true decentralization [5]. Also, many of the cryptocurrencies struggle with finding true utility.

MetaSafe overcomes those issues by making sure the consensus algorithms can be run using low cost processors by ditching proof of work. Also, utility is built in as a fundamental feature. The MetaSafe blockchain will be able to execute smart contracts [6, 7], reward network validators with MetaSafe utility tokens, and provides an open platform to deploy applications for data security and privacy by sharing the resources of the system running MetaSafe (in this case a router and it's connected devices).

The validators are the backbone of MetaSafe blockchain. To make sure it can be run on standard low cost processors, the MetaSafe consensus algorithm is based on a combination of proof-of-stake [8], proof-of-storage, and proof-of-network-bandwidth. In addition, evidence of ownership of physical hardware (verifiable by a unique hardware ID) is also required.

Having utility built in is also key to ensuring the future viability of MetaSafe. MetaSafe creates a means to offer storage, bandwidth, security, location, and other resources that a mesh router can offer to the community. Since a router is central to all connectivity, the resources that can be offered are as endless as the types of devices that can connect to it. A combination of these resources can be used to create services such as password managers, bandwidth sharing, secure peer-to-peer communications, location services, and more. The MetaSafe ERC-20 [9] token (MST) creates a mean of value exchange between resource providers and consumers of those resources.

3. MetaSafe Components

The MetaSafe network platform is developed using the following basic components:

- *Blockchain:* designed to offer "Decentralization", "Immutability", "Verifiability", "Pseudo anonymity" and "Trust less" features. The blockchain is based on Ethereum but is specifically modified to run on low cost hardware
- *Consensus algorithm*: Proof-of-stake, Proof-of-storage, Proof-of-bandwidth
- *Smart contract*: to enable the decentralized applications
- *Utility tokens:* to incentivize validators and application developers.

- *Validator nodes*: validator nodes are the backbone of MetaSafe network. The simplest way to run a validator node is to set up a MetaSafe enabled router using the MetaSafe mobile app.
- *Beacon chain*: this is the main chain of MetaSafe network. Beacon chain is responsible for electing a validator for the respective slots in the main as well as the shard chains. Beacon chain will keep track of the stake and reward tokens when a block is added. It also imposes the policy for bad actors or eliminates nodes that do not meet the requirements (such as responding in time)
- *Shard chains*: these are the parallel blockchains that work with the beacon chain to achieve horizontal scaling for the MetaSafe network. MetaSafe employs 4 shards- where shard 0 is the main chain and shards 1,2,3 are the additional blockchains. Validators will be randomly assigned to a shard as they join the network and at each epoch.
- *Epoch:* regular interval at which the selection and sorting of validators along with the rewarding and slashing operations will be taken place
- *IPFS*: interplanetary file system will be used to extend the memory needed for on-chain and off-chain storage.
- *Cross Chain Transactions*: MetaSafe supports cross chain and bridge transactions which enables it to interoperate with other chains such as Harmony or Ethereum.
- *Peer2Peer Networking*: enables the nodes to communicate among themselves without any centralized controllers or coordinating systems
- *Developer APIs:* decentralized applications can use JSON-RPC and Web3 APIs to interact with MetaSafe network and the resources provided

4. System Architecture

MetaSafe is a blockchain network that is designed to offer “Decentralization”, “Immutability”, “Verifiability”, “Pseudo anonymity” and “Trust-less” features. These are achieved by running a blockchain that is based on Ethereum but is specifically modified to run on low power devices with a set of unique consensus algorithms.

MetaSafe network has two layers - a Consensus Layer and an Execution Layer. The consensus layer is responsible for handling all the beacon related activities such as coordination with the execution layer, running epoch transactions, rewarding, slashing, and validator selections for the next epoch. The execution layer is the layer where the business use cases such as smart contracts, transactions etc will be taking place.

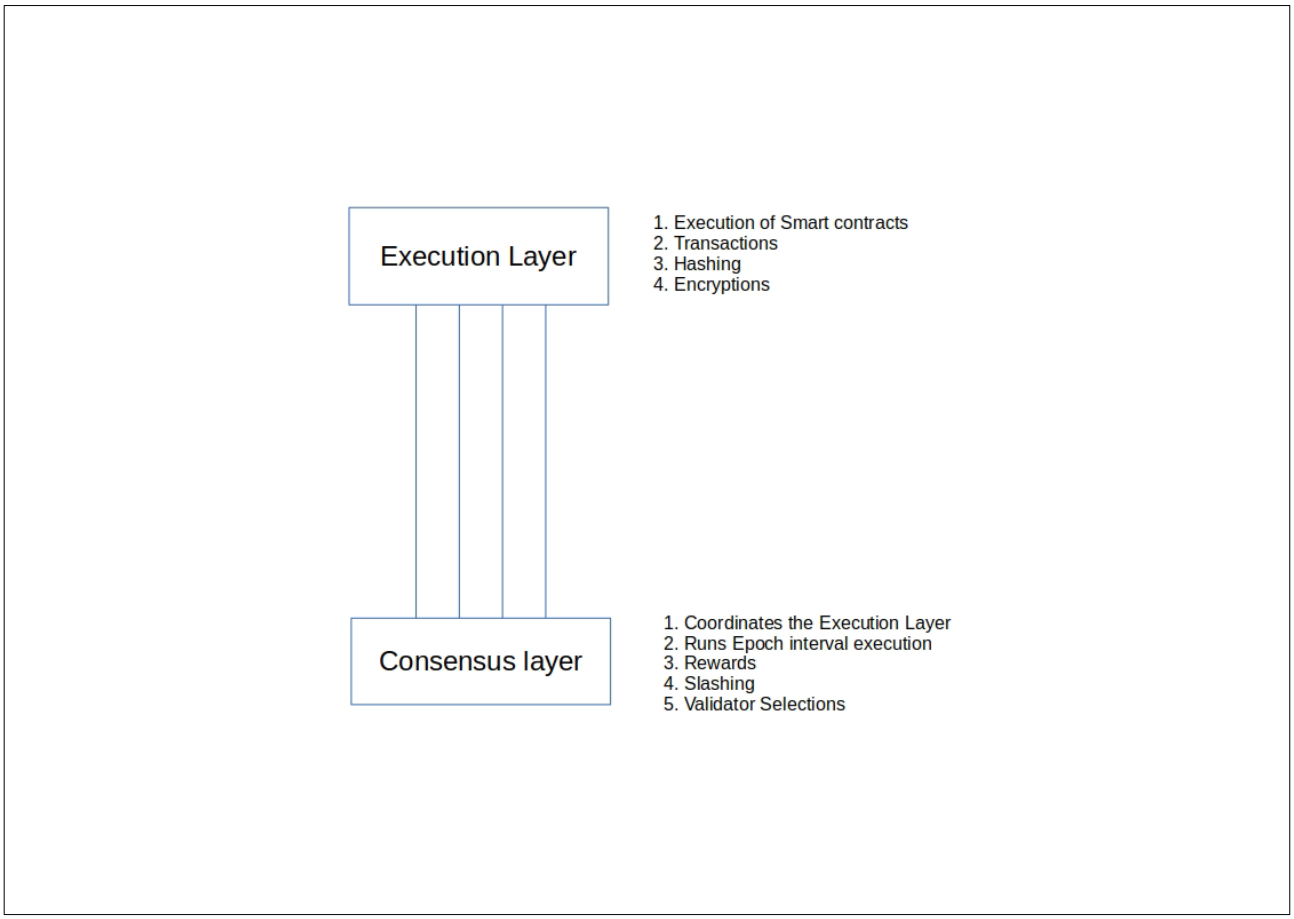


Diagram: Consensus Layer and Execution Layer

5. MetaSafe Blockchain Structure

The MetaSafe blockchain is designed to support 4 shards to optimize cost and performance. Shard 0 is the main chain or beacon chain with three other shards – shard 1, shard 2, and shard 3 as additional blockchains for storage. As validators join the network, they are randomly assigned to a shard. Then at each epoch, the validators will be randomly reassigned to a different shard.

Beacon Chain (shard 0): This is the main chain of the MetaSafe network. The beacon chain is responsible for selecting the validators for the respective slots in the main chain as well as the shard chains. The beacon chain contains the deposit contract in which the validators will be depositing/staking the MST tokens. Beacon chain will keep track of the stake that individual validators has staked on to the network. It will also allocate rewards to the respective proposers when a block is added onto the chain. Further, it will impose the slashing for malicious actors on the network and imposing deductions for the nodes that remain offline for longer periods of time.

Shard Chains (shard 1, shard 2, shard 3): These are the parallel blockchains that work with the beacon chain to enable horizontal scaling on the network database and perform transactions in parallel to achieve higher throughput on the network. MetaSafe sharding implementation is PoS based and removes the need for each node to store the entire blockchain, thus making the blockchain more scalable and secure.

6. Chain Sync Modes

To ensure state and data consistency across the shards, three sync modes are supported by the MetaSafe network.

Main Chain Sync Mode: In this mode of sync, the main chain will be syncing the states between the shard chains. Whenever a transaction or condition needs to update the state in another chain, the data will be sent to the main chain from the origination shard chain and in turn the main chain will perform the sync to the target shard chain.

Client Based Sync Mode: In this mode of sync, the clients which are running the applications will be handling the events and will be sending the state information from one chain to another chain. The client applications act as the bridge between the chains in this mode.

Chain to Chain Sync Mode: In this mode of sync, the shard chains will be directly communicating with the target shard chain to update the state information. Whenever a state change happens on one of the shard chains and this state change must propagate to the next shard chain, the details will be directly sent to the target shard chain via RPC calls.

7. Consensus Algorithm

The consensus algorithm [10, 11, 12] is the main component of any blockchain. The algorithms decide how quickly and securely the network can reach a consensus to commit the next block in the chain. In the traditional proof-of-work (PoW) consensus process, validators compete to find the solution to a cryptographic puzzle. The winner gets to propose the next block and earns tokens as rewards. In this consensus protocol, the assumption is that more than 50% of the validators are honest nodes. However, this proof has all the issues listed above that make it not suitable for MetaSafe.

MetaSafe uses the Practical Byzantine Fault Tolerance (PBFT) consensus algorithm that allows the network to reach a consensus even when a small number of nodes are faulty or dishonest. During the transmission, PBFT uses cryptographic algorithms, such as hash, to ensure that the information is immutable and unquestionable. PBFT brings down the computational complexity from exponential to polynomial. Using PBFT, a distributed system that constitutes 3f+1 nodes (f represents the number of faulty nodes or byzantine nodes), a consensus can be achieved successfully and honestly as long as no less than 2f+1 non-byzantine nodes are functioning normally.

Each time the consensus protocol runs, it starts with the beacon chain electing one node as the "leader" and the rest of the randomly selected nodes as validators. Each round of consensus protocol consists of two phases: 1) the preparation phase and 2) the commit phase.

In the preparation phase, the leader sends the proposal to all the validators. Each of the validators in turn sends their votes to everyone else. This ensures that all other validators count the votes of each other validator. The preparation phase completes when more than 2f+1 votes are counted as consistent (where f is the number of faulty validators, and the total number of validators plus the leader is 3f+1) by each validator including the leader.

The commit phase is a similar vote counting process where the next block is created if consensus is achieved. Here also, the leader broadcasts the decision to all validators, who in turn rebroadcast the message to everyone else. When each validator and the leader see 2f+1 consistent votes the commit phase is completed.

Since MetaSafe consensus uses proof-of-stake (POS), the consensus algorithm is modified by allowing validators to have voting power that is proportional to their voting shares gained by staking. So, instead of counting for at least 2f+1 signatures from validators, the leader counts the signatures from the validators that effectively make up at least 2f+1voting shares.

But there is a problem with PBFT as it is. The rebroadcasting of votes to and from all the validators results in too many transmissions O(N2) and complexity becomes not sustainable for a blockchain network with a large number of validator nodes. MetaSafe implementation avoids the rebroadcasting of PBFT by all validators. This is achieved by adopting BLS (Boneh-Lynn-Shacham) multi-signature protocol, a variant of PBFT. BLS is a pairing-based cryptographic signature scheme that supports multiple validators signing their vote with a constant-sized signature.

The BLS-modified PBFT consensus protocol works as follows:

1. The leader proposes the new block and sends the block header to all validators.
2. The validators validate the block header, sign it with a BLS signature, and send the signature back to the leader without rebroadcasting to all validators.
3. The leader waits for at least 2f+1 (as per PBFT algorithm) consistent signatures from validators and aggregates them into a BLS constant-sized, multi-signature.
4. The leader sends the aggregated multi-signature to all validators
5. The validators confirm that the multi-signature has at least 2f+1 signers, sign the received message, and send it back to the leader.
6. The leader waits for at least 2f+1 valid signatures from Step 4 and aggregates them together into a BLS multi-signature.
7. Finally, the leader commits the new block with multi-signatures and sends the new block to all validators.

MetaSafe's implementation of BLS-modified PBFT features the following fault and attack tolerance mechanisms:

1. *Sybil Attack:* in a Sybil attack, one entity assumes many identities and tries to gain 51% control over the network. To prevent Sybil attacks, various techniques are present such as "Identity Validation", "Social Trust Graphs", "Personhood Validation", "Economic Costs" etc., MetaSafe uses PoS Proof of Stake to prevent Sybil attacks. In MetaSafe, if a node wants to be a validator, it has to stake a certain amount of tokens. As the number of nodes in the network increases, to control the network or gain command over the network, the attacker has to stake enough tokens with multiple identities to exceed more than 51% of the shares staked. This becomes prohibitively expensive for attackers.
2. *Faulty Leader Node:* if a leader node is a faulty node where it is trying to stall or corrupt the communication, consensus will not be reached or reached in time. The faulty leader will be ignored, and a new leader will be selected to complete the task.
3. *Faulty Validator:* let's assume that in a committee, the validator node is not proposing a solution in time or corrupting the data, then as per PBFT, if at least 2f+1 replies are coming from honest nodes, the algorithm will continue to arrive at a consensus. In a situation where there are more faulty nodes and the consensus is not reached, this view will be discarded, and the algorithm will move on to select new validators.
4. *Retransmission Node Loading:* during the PBFT phases, the messages generated at each phase must be transmitted between leader and replica to arrive at a solution and vice versa. If

faulty replicas are retransmitting the messages to overload the network, the other replicas can detect this by checking the timestamp and BLS signature. They will simply choose to ignore the message if there is a timestamp duplication or if the BLS signature is invalid.

5. *Distributed Validator Across Shards:* In MetaSafe, validators are not always assigned to only one shard. At each epoch, the validators are randomly selected and grouped to form proposers and validators with committees for each shard until another epoch. This prevents faulty or malicious validators from taking control any one shard.
6. *DDoS Attack:* To prevent or address DDOS attacks, a transaction fee is present in the MetaSafe, where every transaction that alters the state of the blocks in the MetaSafe network must be done with a transaction fee associated with it. The attacker sending this must pay the transaction fee without which the receiving nodes will reject the transaction and will not allow the transaction to propagate onto the network for finality. The transaction fee creates a cost barrier to preventing DDOS attacks.
7. *Eclipse Attack:* a node in a network of nodes needs to connect to a subset of nodes to sync the blocks and view its ledger. If the attacker can control the connecting nodes, then the node trying to sync its blocks can be presented with completely different information than the correct ones. MetaSafe prevents the eclipse attack by regularly generating the Kademlia routing table at certain time intervals which will randomise the nodes with which the current node is communicating with. Due to this randomisation, eclipse attacks are at most temporary and can be recovered in the next interval by verifying the chain data.

8. Distributed Randomness [13]

At every epoch, certain numbers of validators are assigned to a shard to create the next block. The validators are picked randomly based on a protocol that is unpredictable, unbiased yet verifiable, and scalable. The first two characteristics are obvious - no one should be able to predict the random number or influence the generation of random numbers. It is also important that the validity of the random number is verifiable by anyone, and the solution should work when large numbers of validator nodes are present.

MetaSafe utilizes Verifiable Random Function (VRF) and Verifiable Delay Function (VDF) to implement Distributed Randomness Generation (DRG) algorithm to pick the validators [14, 15, 16, 17, 18, 19, 20]. The entire system works by selecting a leader at random and the leader will construct an initial message with a hash of the previous block. This message will be sent to all the validators. After the validator receive this message, VRF is computed to create a random R and the proof P as follows:

(Ri, Pi) = VRF(Ski, H(Bn-1), v)

where

i – validator node that is computing the VRF
Ri - random value from i validator node
Pi - Zero Knowledge Proof (ZKP) from i validator node
VRF- Verifiable Random Function
Ski – Private key of the i validator node
H(Bn-1) – SHA3 of the previous block
v - the current view number of the current epoch.

After the validators generate the random value, the leader will wait for f+1 replies from the validators, where f is the total number of faulty nodes according to the PBFT algorithm. Once the responses are received from the validators, Prnd will be computed as follows:

Prnd = XOR(r0, r1, r2, r3, ... rn) where n = 3f + 1 nodes and the total replies are f+1

The leader then runs PBFT among all the validators to reach a consensus on the Prnd and commit the Prnd in block n. After Prnd is committed, the leader will start the actual randomness computing Rnd = VDF(Prnd, T). T is the difficulty and is set algorithmically such that randomness can only be computed after a random k number of blocks. Once Rnd is calculated, the leader initiates PBFT among all validators to agree on the validity of Rnd and commit the Rnd into the chain.

VRF has three stages

1. Keygen(r): (Pk, Sk) where Pk is the public key and Sk is the private key using BLS key generation algorithm
2. Compute(Sk, M): Hash(Sign(Sk, M)) -> (R, P) where R is the hashing of the signature signed with Private key Sk of message M and P is the VRF proof which is equal to the signature from Sign(Sk, M)
3. Verify(Pk, M, R, P): SignatureVerify(Pk, P, M) = True & Hash(P) = R

VRF is achieved via a precompiled smart contract present in the chain
VDF where delay is added to a Proof of Stake consensus algorithm to prevent malicious validator from predicting the randomness and influencing the decision of mining a block.

9. Epoch Interval and Process

MetaSafe epoch interval is the time interval at which certain tasks such as adding new validator nodes to the chain, reassignment of the validators to each shard, and incentives are paid out. Epoch time is selected to be every 14400 slots. With 6 seconds per slot, this will be 24 hours. This is defined at the genesis block and will be fixed for a duration of a month. After the first month, the current validators can propose a new epoch time interval. If a majority of the validators (N/2+1 validators) vote for a new epoch time interval, then the new epoch time interval is accepted and will take effect immediately in the next cycle. If the number of votes does not reach N/2 + 1 then the epoch interval time will remain as is for the duration of one more month and will again be eligible for voting after one month has passed. [21]

During the start of each epoch, new validators will be added to the network and assigned to a shard. The existing validators will be reassigned to their new shard for this epoch. The assignment of the validators will be done based on a distributable random number generator DRN function. In this process, a leader validator node will be selected based on the current leader selection algorithm. After this process is done, the leader will broadcast two large unsigned integer values M and N, where M is less than N and M is not equal to N, to all the nodes. After all the validators receive these two numbers, they will pick a random number X. The validator node will calculate the hash value for verification purposes using its private key and public key as the hash. Hash(V-Rand, Private Key) and Hash(V-Rand, Public Key). These values along with the validator's public key and a random number will be broadcasted in the network. The leader node will be waiting for a predefined time and after that, the leader will broadcast the slot ranges into which the random numbers will be residing and will commit this information into the block along with the random numbers chosen by the validators. Once these values are committed then the validator nodes will check the respective ranges committed in the block and will join the appropriate shard.

The example of this entire process is given below

1. After the leader is selected, the leader will propagate the M and N values 100000, 200000
2. Consider 3 validator nodes are being categorised to three shards, V1, V2, V3
3. Each validator node chooses a random number between M and N as follows
    1. V1 – 143345
    2. V2 – 195534
    3. V3 – 159443
4. Along with Random values, two more values are generated as follows

    1. Hash(V1(Rand), Private Key)
    2. Hash(V1(Rand), Public Key)
5. The resultant is broadcasted to the entire network as follows
    1. [V1 public key, Hash(V1(Rand), Private Key), Hash(V1(Rand), Public Key), V1 Rand]
6. The leader proposes the following slots for 3 shards
    1. 120000-150000 – Shard 1
    2. 160000- 200000 – Shard 3
    3. Rest of numbers – Shard 2
7. In this process the Validator nodes V1, V2 and V3 are assigned as follows
    1. V1 – Shard 1
    2. V2 – Shard 3
    3. V3 – Shard 2
8. This allocation can be verified by anyone using Hash(V1(Rand), Public Key) and V1(Rand)
9. If someone joins the wrong shard then it can be easily detected with step 8

Fault Tolerance:

1. In case the leader node does not select the slot ranges within the predefined time interval, then the entire process will be reinitiated by selecting a new leader and imposing the penalty on the previous leader node by taking away the staked tokens.
2. If any of the validator nodes modify the secret random number after the broadcast is done, then it can be easily detected with the block data containing the secret random numbers from the validators. If this is detected, a penalty will be imposed on the respective validator node by taking away staked tokens
3. If the validator nodes do not broadcast the random numbers within the time interval defined in the network, then the validator node will be removed from participating in the next epoch interval. Tokens will not be taken in this context because the node might be temporarily experiencing network issues or other unforeseen issues. The node can participate again at the start of the next epoch.

At each epoch, before the validators are assigned their shard, MetaSafe incentives are paid out to each of the nodes based on the tokenomics defined in section 12. The leader node selected to perform the epoch operation will also pay out the gas fees collected for each transaction to the validator nodes that validated those transactions. These will all be in the form of MetaSafe utility tokens.

In addition, at each epoch, a majority of the actual fiat collected for applications built on top of the MetaSafe blockchain will be credited to the validators in proportion to the amount of MST they staked to the total MST staked on the MetaSafe network. For example, if Bobby staked 1000 MST to the network and there are 10,000 MSTs total staked across all MetaSafe network then Bobby will get 10% of the fiat collected in that epoch as a credit to his fiat wallet.

10. Validator nodes

Validator nodes are the main workhorses of the MetaSafe network. These are the nodes operated by the community of validators running and maintaining the MetaSafe network, providing peer-to-peer connections and secure storage. These are the computing systems that help in proposing the new blocks, selecting the validators at epoch, enforcing the slash rules, maintaining the database of the MetaSafe network, offer peer to peer connectivity, performing cross-chain transactions, executing the smart contracts, etc.,

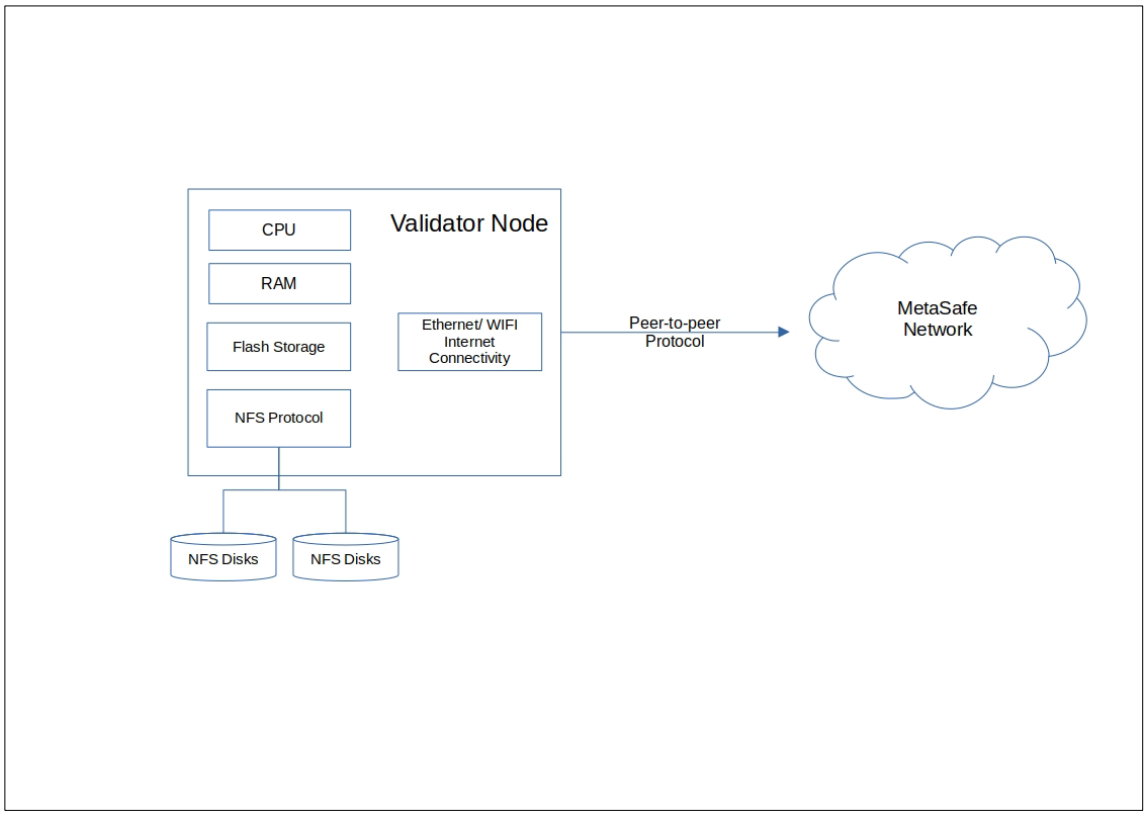


Diagram: MetaSafe Validator Node

Protocols: MetaSafe network is a decentralized network that is designed to run on low cost processors that communicate among themselves using certain protocols.

1. For the network communication MetaSafe uses Peer-to-Peer mechanism where the nodes will do discovery by connecting to boot nodes or beacon nodes.
2. MetaSafe network nodes will work on both IPv4 and IPv6 with transport protocols supporting over TCP and UDP.
3. Decentralized Apps (DAPP) will interact with the MetaSafe Network using Web3 or JSON RPC protocols.
4. DAPPs can also interact with the network using HTTPS connections and web socket connections if the node is configured to serve these connections.
5. IPFS protocols is used to store the data in IPFS drives and extend the storage in a random and distributed fashion for off-chain storage
6. NFS protocol is used to extend storage for on-chain storage

Storage: Blockchain requires storage. For MetaSafe, the storage can be provided onboard or offboard. Gryphon 6E routers have 8GB of eMMC storage and has the minimum needed to be a MetaSafe validator. To extend the on-chain storage, MetaSafe supports NFS for external storage solutions. The validators can also use IPFS to provide additional storage in the form of off-chain storage. Only encrypted references to the off-chain storage location and its meta data are stored on the MetaSafe blockchain. Off-chain storage allows for information to be stored in a more efficient manner than on blockchain.

11. MetaSafe Network

11.1 Node Routing Protocol

Peer-to-peer networks can communicate with each other with a set of protocols. In the first generation of peer-to-peer file networks, a central database was used to coordinate the network, in the second generation of peer-to-peer networks, network flooding is used to locate the files. In third generation networks, distributed hash tables were used to locate the files. Kademlia routing [22, 23] uses the distance between nodes by using XOR of node IDs and taking the resultant as an unsigned integer number. The distance is not geographical, but rather the distance between the UUID of the nodes. XOR operation yields the following properties: a) distance between the node and self is 0. b) the distance between nodes A and B is the same as B and A, so the distance calculations are symmetric. c) it has triangle inequality. The advantage of using Kademlia protocol is the ability to reach any node in the network quickly and efficiently without flooding the entire network.

Kademlia network with 2n nodes will only take n steps (in the worst case) to find another node. Kademlia is a fixed-size routing table. If the node id has 128 bits of UUID then the node will be maintaining a list of 128 entries. These entries can contain UUID, IP address, and port of other nodes. Taking an example of 3 bits for UUID generations, the following table will be generated

Node UUID	2^0 - 000	2^1 – 010	2^2 - 100
0	1	2	4
1	0	3	5
2	3	0	4
3	2	1	7
4	5	6	0
5	4	7	1
6	7	6	2
7	6	5	3

In the above routing table, we can see that node 0 neighbours are node 1, node 2 and node 4, likewise, for node 3 the neighbours are node 2, node 1 and node 7 and so on. Here for illustration, the entire table is shown. In the diagram below, we can see a different set of routes that are generated in the peer-to-peer network without any central authority. In the below example, if node 3 wants to send some traffic to node 6, then the paths generated are “3 – 7 – 6” or “3 – 2 – 4 – 6”.

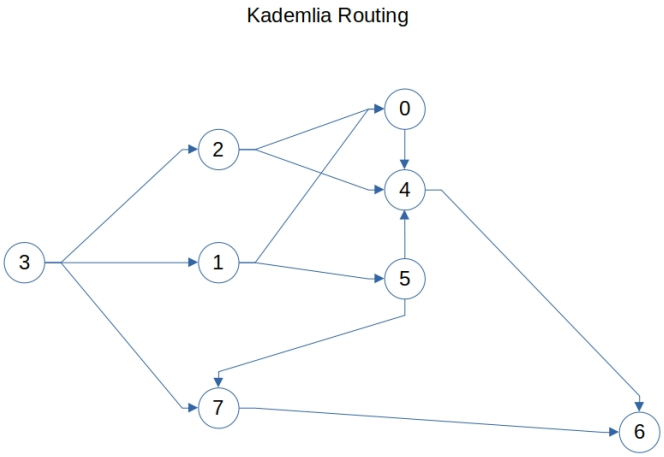


Diagram: Kademlia Routing

11.2 Proofs

MetaSafe operates on a modified proof of stake consensus protocol in which the user must stake MetaSafe tokens (MST) along with storage space and bandwidth to become eligible as a validator.

Proof of Stake

The proof of stake algorithm is an algorithm where the validators will be selected at random depending on the amount of MSTs staked instead of burning energy to solve a cryptographic puzzle. The staking contract is present on the beacon chain where the validators will be depositing the tokens and participating.

Proof of Storage

In addition to staking MSTs, the validator must also offer storage space for storing the blockchain. The system has to offer at least 8GB of storage to be considered a suitable validator. This will be checked at each slot.

Proof of Bandwidth

Another requirement to be a validator is the amount of bandwidth that is available to the system. The minimum bandwidth that's required is 100mbps down and 10mbps up. This will be checked at each epoch.

Consensus Process:

The beacon chain is the core of the MetaSafe network. Its main function is to store and manage the registry of validators. Validators are the nodes that have staked a minimum amount of tokens to the deposit contract deployed on the MetaSafe beacon chain and have enough storage and bandwidth.

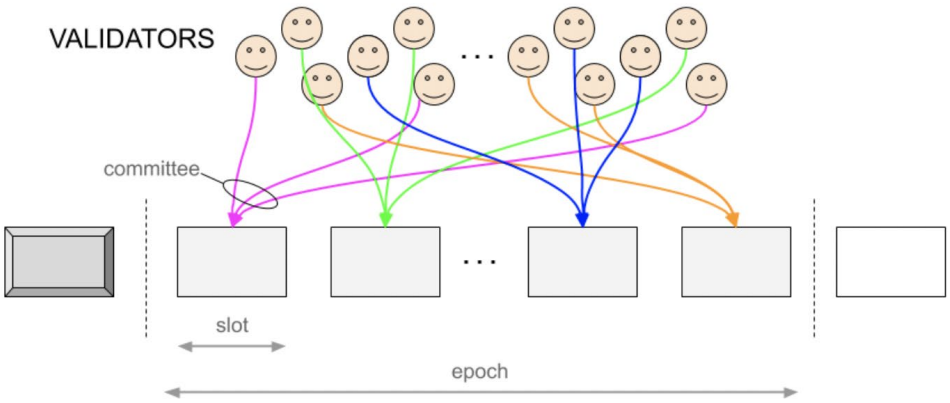
The purpose of the Beacon chain with validator registry is as follows:

1. Assign validators
2. Finalize checkpoints
3. Perform random number generation
4. Progress beacon chain
5. Link and vote on the transactions in shard chains

A slot is a set time frame for proposing the next block (in MetaSafe, a slot is 10 secs). A block is the storage space containing transactions, data, and smart contracts. A slot may or may not have any blocks to process. One epoch contains 14400 slots.

After each epoch, the available validators are randomly selected and merged to form new committees. One or more individual committees are required to attest the blocks. There are 127 committees and 1 proposer. The proposer is the node that proposes the block and the rest of the 127 committees containing one or more validators will attest the block. The process of selecting the validators

and committees lies with the beacon chain. It is responsible for randomly assigning the proposer and committees for each slot.



12 MetaSafe Tokenomics

MetaSafe network is powered by the MetaSafe utility token which is an ERC-20 compliant token. The MST is used for staking and to incentivize the build out of the MetaSafe network. The total number of MST tokens available is capped at 10,000,000,000 tokens. The MST tokens can be divided into 18 decimal numbers for fractional transactions.

The tokens will be allocated as follows:

- 40% treasury reserve
- 40% used to build out network (airdrops, stake, and ongoing rewards)
- 10% for fostering ecosystem development(3rd party apps, grants, marketing, etc)
- 10% for MetaSafe team and developers

MetaSafe validators will earn a 10% annualized interest on staked MSTs paid out at each epoch. Locked MSTs for staking will be airdropped to each validator with decreasing amounts as the network is built out. The amount of locked MST airdropped will also be proportional to the bandwidth and memory committed by the validators.

Profit sharing is an important part of the MetaSafe utility token. 70% of the revenue collected from apps using MetaSafe, such as the password manager, will be redistributed back out to the MetaSafe validators (you) in proportion to the amount of your staked MST.

A DAO or multiple DAOs will be created in the future to set the interest rate for staked MST, determine max staking and minimum staking amounts, and among other things, determine how to allocate or grant the MSTs reserved for ecosystem development.

AirDrop Schedule:

Number of Connected Nodes		MST AirDropped	Total Dropped
0	2,500	30,000	75,000,000
2,501	10,000	15,000	112,485,000
10,001	50,000	10,000	399,990,000
50,001	100,000	7,500	374,992,500
100,001	250,000	2,500	374,997,500
250,001	750,000	1,250	624,998,750

Total Air Dropped at full network build out of 750K nodes is 1,962,463,750 MSTs. The remaining incentive pool for interest and ongoing rewards will be 2,037,536,250 MSTs.

The reward for staking more storage beyond the internal router storage will have the following schedule:

- 0.003 MST per epoch per GB up to 250GB
- 0.002 MST per epoch per GB up to 500GB
- 0.001 MST per epoch per GB up to 1T
- 0.0005 MST per epoch per GB up to 2T

The incentive pool is expected to be fully exhausted in 7-10 years. Once that happens, the network will be sustained by the revenue sharing from the various apps developed on MetaSafe and the gas fees charged for appending to the blockchain.

Maximum staking per node is 50,000 MST. Minimum stake to be a validator is 1250 MST.

13. Ongoing and Future Work

Here we list some of the applications that may be developed using MetaSafe.

Password and Identity Management (currently under development)

- Decentralized secured storage on blockchain
- Passwords, Accounts, Credit cards, ID, Medical records
- Browser plugins and mobile apps to make it easy

Health Data Management

- Decentralized data storage for patient

- Stores all health records, diagnoses, lab data
- Can be shared with health providers as and when required

Truly Private Peer-to-Peer VPN

- Decentralized VPN using tor-like techniques
- Gryphons become secure relays for traffic
- Private and secure

WiFi Bandwidth Sharing

- Allow anyone to access and pass data on the WiFi network by paying a small fee using MST

14. References

1. Web2 verses Web3, https://ethereum.org/en/developers/docs/web2-vs-web3 [Online]
2. Mordor Intelligence, “https://www.mordorintelligence.com/industry-reports/password-management-market,” [Online].
3. Ethereum Blockchain, https://ethereum.org, [Online]
4. Bitcoin Consumes More Electricity Than Argentina, https://www.bbc.com/news/technology-56012952 [Online]
5. I. M. &. A. Schoar, “Blockchain Analysis of the Bitcoin Market,” NBER, 2021.
6. Smart Contracts, https://www.openzeppelin.com [Online]
7. Smart Contracts, https://www.ibm.com/topics/smart-contracts, https://ethereum.org/en/developers/docs/smart-contracts [Online]
8. Proof of Stake, https://ethereum.org/en/developers/docs/consensus-mechanisms/pos [Online]
9. ERC-20 Tokens, https://ethereum.org/en/developers/docs/standards/tokens/erc-20 [Online]
10. Practial Byzantine Fault Tolerance, https://www.nature.com/articles/s41598-022-08587-1 [Online]
11. PBFT, https://coredevs.medium.com/an-introduction-to-pbft-consensus-algorithm-11cbd90aaec [Online]
12. PBFT, https://pmg.csail.mit.edu/papers/osdi99.pdf [Online]
13. Distributed Random Number Generation for the Need of Public Governance, https://arxiv.org/pdf/1803.05385.pdf [Online]
14. Verifiable Random Functions, https://medium.com/asecuritysite-when-bob-met-alice/verifiable-random-functions-4563d6eb17ab [Online]
15. Verifiable Random Function, https://en.wikipedia.org/wiki/Verifiable_random_function [Online]
16. Verifiable Random Function, https://open.bu.edu/bitstream/handle/2144/29225/draft-irtf-cfrg-vrf-01.pdf [Online]
17. Verifiable Random Function, https://dash.harvard.edu/bitstream/handle/1/5028196/Vadhan_VerifRandomFunction.pdf [Online]
18. VRF, https://wiki.polkadot.network/docs/learn-randomness [Online]
19. VRF, Solidity: https://github.com/witnet/vrf-solidity [Online]
20. Harmony VRF, https://medium.com/harmony-one/introducing-harmony-vrf-4fc51e175c2 [Online]
21. EPOCH, https://www.bitdegree.org/crypto/learn/crypto-terms/what-is-epoch [Online]
22. A Brief Overview of Kademlia, https://medium.com/coinmonks/a-brief-overview-of-kademlia-and-its-use-in-various-decentralized-platforms-da08a7f72b8f [Online]
23. Kademlia, A Peer to Peer Information System Based on XOR: https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf [Online]