Razor Network: A truly decentralized oracle platform with onchain governance

Hrishikesh Huilgolkar
CTO, Razor network
Revision: 30th June 2019
Early draft for feedback. V 0.4

Abstract

Decentralized technologies such as blockchain are revolutionizing many industries including finance. Applications such as decentralized lending, stable currencies, prediction markets and synthetic assets are being researched and built on top of them. Many such applications depend on real world data, which is not readily available inside the blockchain environment due to their design. Currently this problem is being solved by something called an "Oracle", which is an entity which reads real world data and feeds it to the blockchain. Current Oracle solutions are either centralized, or vulnerable to certain attacks due to the lack of on-chain[1] governance or due to low economic security. Current oracle solutions may work short term but are not suitable for long term applications, which is essential in decentralized applications.

In this paper we propose a fully decentralized oracle network called "Razor network" with built-in on-chain governance, so that the network can thwart such attacks and remain functional in a constantly evolving environment. Razor network is resilient to bribing attacks since it utilizes high degree of redundancy and offers high economic security for all applications regardless of the fees being paid to the oracle.

Razor network consists of three layers: Validation, Governance and Application. (1) Validation layer consists of stakers[2] who accept tasks from a job queue, perform fetching of information from real world, process and aggregate the results and serve them to the requesting contract. (2) Governance layer makes decisions on whitelisting and blacklisting various data-feeds and curating collections of them for various use cases. (3) Application layer consists of various applications that will use this service for various use cases.

Razor network uses a proof of stake consensus algorithm and uses a native utility token called Schell (SCH). Schells are needed to be locked to participate as a staker in the network. SCH are necessary to use this network to perform tasks. Stakers are awarded these fees as well as block rewards for participating in the network. The amount of staked tokens and on-chain reputation of the staker determine their influence in the network.

The design goals of the Razor network are to ensure the long term sustainability of the oracle and the data feeds it provides, high degree of decentralization, high economic security and protection of both stakers and clients of the oracle. We will be utilizing Ethereum network in the initial version and explore supporting other blockchain platforms in future.

---

[1] on-chain means the decisions are performed by a smart contract taking into account various factors

[2] Stakers are users who lock their funds in a smart contract. This action is known as "staking". Stakers are expected to perform their duties honestly, or else they may lose their funds and future profits.

Table of contents

List of figures

(this page is manually updated and may be out of date)

1 Introduction

1.1 Motivation

Decentralized networks, through the use of smart contracts, are disrupting established systems by removing the need for intermediaries and opening up equal access to everyone. Finance might be the most impacted sector: some of the examples of decentralized finance (also known as Defi) applications include:

1. Decentralized stable currencies, also known as "Stablecoins"
2. Decentralized Insurance
3. Decentralized Prediction markets
4. Decentralized Creation of Synthetic assets
5. Decentralized exchanges and derivatives trading market
6. Decentralized Identity

These applications consists of a set of smart contracts deployed inside a blockchain. Such applications often require data from outside the confinements of the blockchains they reside in. Blockchains, being a deterministic system, only depend on the information available inside the system, as that is the only information that can be cryptographically verified by all participants. Blockchains do not readily have access to the outside world, by design.

Hence, to facilitate the access to the outside world, concept of "Oracles" has been proposed. An Oracle is an entity which queries the required data from the outside world and feeds it to the blockchain. Traditionally, this has been attempted through the use of trusted intermediaries. They typically facilitate this by accessing a data feed through an API or a webpage, validating it through multiple sources and feeding it to the blockchain. These intermediaries are centralised entities and hence, introduce single point of failure in a decentralized system. Such weaknesses are not desirable because they reduce the utility and security of a decentralized system to that of a centralized, trusted one.

To combat this weakness, the concept of a decentralized oracles was introduced.
In this paper we propose a general purpose, resilient, decentralized and trustless[3] Oracle platform, which addresses various shortfalls in the current designs.

1.2 Previous work

Previous attempts to solve this problem include application specific oracles such as Augur, gnosis, MakerDao, centralized oracles such as Provable and general purpose decentralized oracle platforms such as Truthcoin, SchellingCoin, Chainlink and Witnet. The current work is inspired by SchellingCoin and Truthcoin protocols.

Developing a decentralized oracle is deemed a challenging problem. This is due to the possibility of multiple kinds of attacks such as collusion, takeover, griefing etc., requirement of subjective and objective decision making, determining the "truth", and also due to the technological limitations of the underlying blockchain protocol. Current general purpose oracle platforms face the following issues:

1. Lack of high degree of decentralization and economic security
2. Lack of long term viability
3. Cognitive load on application developers
4. Targeted misinformation attacks

[3] Trustless here means that no trusted third party or intermediaries are needed

1.2.1 Lack of high degree of decentralization and economic security

Some of the current solutions involve a trusted centralized mediatory, which acts as a single point of failure, while others combine results from a few stakeholders of the network. Often, if a high degree of decentralization is desired, the client has to pay high amount of fees proportional to the degree of decentralization desired. This means that the accuracy and economic security of the oracle platform is not the same for all jobs, and the oracle cannot be trusted as the "Universal source of truth".

Let's explore this problem with an example. Assume there is a CDP[4] backed stablecoin project called "Acme". Acme platform issues US Dollar-pegged stablecoins backed by ether on the Ethereum blockchain, and hence, requires a data-feed of ether/USD. Acme depends on a decentralized oracle platform called "Truthbox". Truthbox assigns stakers to the query and reports the ether/USD price, everytime Acme requests the data with a fee. The number of stakers assigned by Truthbox depends on the amount of fees being paid by Acme.

This shows the weakness of the system. If someone requests to report the price to Acme with a very low fee, Truthbox will likely assign the task to a single staker (or very few stakers). Hence the system reduces to a centralized or semi-centralized oracle. The protocol, in such cases, becomes vulnerable to various attacks such as griefing, bribing and collusion.

If the oracle reports a price which is too far from the actual price, it can cause a large amount of liquidations and instability of the entire Acme platform. Hence it is required for Acme to pay large amount of fees everytime it requests a price from Truthbox, to make sure there is sufficient decentralization and economic security. But it still leaves it open to attacks where the attacker pays insignificant amount of fees to report an inaccurate datapoint to Acme.

1.2.2 Lack of long term viability

Current general purpose oracle platforms are not suitable for long term applications. They are data-feed-centric. In case the data feed is compromised or becomes defunct, the oracle service becomes dysfunctional.

Some oracle platforms such as Chainlink are marketplace based, where a decision needs to be made to select oracle providers with higher reputation everytime a data point needs to be fetched, as the set of oracle providers and their reputation is constantly changing.

Making decision on choosing the data feed and the oracle providers requires constant verification and decision making. This decision making cannot be made by a smart contract autonomously and requires decisions to be made by the stakeholders of the application. And hence, due to the constantly changing nature of the world outside blockchain, the current oracle solutions are not viable for long term applications.

Due to the lack of on-chain governance in current oracle platforms, the burden of implementing on-chain governance falls on the shoulders of application developers. It also adds cognitive load on the stakeholders of the applications.

1.2.3 Cognitive load on developers

As we discussed in the above section, the burden of balancing between fees and economic security falls on the shoulders of the clients or the application developers. In some platforms, developers are given the flexibility of choosing incentivisation and punishment

[4] CDP means Collateralized Debt Position

mechanism, aggregation method, etc. While this is desirable in some applications, incorrect decision making by the application developers can cause serious issues.

1.2.4 Targeted misinformation attacks

Oracle platforms are vulnerable to selective misinformation attacks. In this attack, the attacker asks the oracle to report a value from a URL she directly controls. She can then program the website to report different data on each request. The attacker may even chose to report different values to 5% or 10% of the requests.

Since most oracle providers use Truth-by-Consensus algorithms, this can cause reputational or financial loss to the stakers even though they were acting honestly.

1.3 Design Goals

Design decisions have been made with the following goals in mind:

1. High Economic security
2. High degree of decentralization
3. Long term viability
4. Protection of stakers from various kinds of attacks
5. Protection of clients[5] from malicious stakers
6. Censorship resistance
7. Ease of use for developers

1.3.1 High economic security

Economic security is simply the amount of financial resources required to compromise a network. Any decentralized network can be compromised with high enough financial resources. However, if the financial benefits of compromise are less than the cost, it is unprofitable to attack the network, hence unlikely that anyone will attempt to do so.

Providing high economic security is one of the biggest design goals for this protocol. There is a clear need for an oracle protocol which provides this feature. Providing high and calculable high economic security provides guarantees for applications to be secure until a certain degree of economic value.

Razor provides same economic security to all requests regardless of the fees being paid. Providing maximum economic security within technical limits is attempted. More details are provided in 2.5.1

1.3.2 High degree of decentralization

Blockchain is sometimes referred to as a “trust machine”. This is because blockchain removes the need of trusting intermediaries and allows a platform to do peer to peer transaction without counterparty risk. Many decentralized applications are taking advantage of this property to build decentralized financial applications.

It is essential to for oracle platforms to have a high degree of decentralization. Which means that in order to compromise the network, large amount of entities need to be compromised.

To achieve this, Razor uses a proof of stake network where large amount of individual stakers can participate. Razor acts as an abstraction layer between clients and stakers so that any stakers can join and leave the network without having any effect on the client

[5] Here, clients are entities who are requesting data-points from our oracle

applications. Game theoretical and cryptographic measures such as commit-reveal scheme and inability to use staking pools provides further collusion and centralization resistance.

1.3.3 Long term viability

Razor network has been designed to be viable for long term applications. Typical decentralized applications have smart contracts which rely on external data. The oracle provider is hardcoded in those smart contracts cannot be changed easily without centralization or without complex governance process, hence the oracle service needs to be available for long term.

The data feeds themselves are usually centralized and may stop providing service or get compromised. Hence decentralized applications cannot depend on them for long term applications.

Razor network features its own governance layer which whitelists reputable data feed sources and blacklists them in case they are compromised or become defunct.

Razor network also has capability to create collections. Collections are set of curated data feeds for the same data.

1.3.4 Protection of staker from various attacks

We propose a governance layer to protect the interests of stakers. Governance layer can make decisions on which data sources can be served via the platform to protect themselves from malicious data sources, selective misinformation attacks, undecidability and unverifiability, etc.

1.3.5 Protection of clients from malicious stakers

An influence based consensus protocol is introduced in Razor to punish malicious stakers and takes away their ability to influence network and earn rewards. This protects the clients from malicious stakers who may try to report incorrect or inaccurate data points in order to influence the result.

1.3.6 Censorship resistance

A censorship attack is one where the actions of users, such as stakers and clients, are censored maliciously to achieve desired results. Layer-2 scalability solutions such as Plasma lack censorship resistance since the operators are able to censor any transaction. If such an attack occurs, it may cause temporary disruption to the applications relying on them.

Due to these reasons, we have decided to use the Ethereum Mainnet as the base platform, as it is currently the most censorship resistant smart contract platform available. Layer-2 scalability solutions will be explored in the future provided they are able to provide sufficient decentralization and censorship resistance features.

1.3.7 Ease of use for application developers

In Razor platform, decisions such as choosing level of economic security, aggregation function, evaluating reputation of stakers and selecting stakers, etc. have been abstracted away for the benefit of developers. Developers can easily and safely use integrate Razor platform without knowing the underlying architecture.

1.4 Architectural overview

Razor network consists of 3 layers:

1. Validation layer
2. Governance layer
3. Application layer

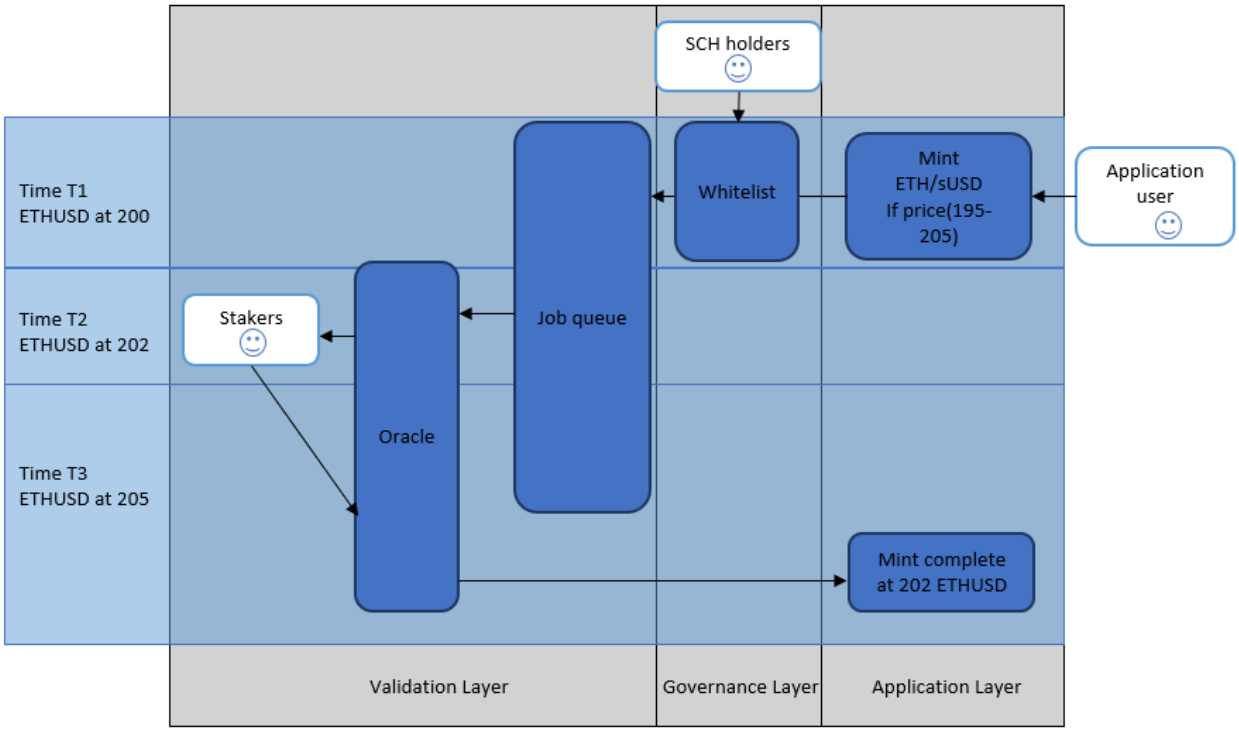

Figure 1: Architectural overview

1.4.1 Validation layer

Validation layer consists of the job queue and the oracle cycle. The actors are the stakers who process jobs in the job queue and provide the result to the contract as requested.

Stakers must deposit their Schells to become a staker in the oracle platform. They process top jobs in job queue in batches of ***J***. The stakers query the URL as mentioned in the job specifications and perform required data processing operations on it to before submitting it to the oracle contract. Aggregation is then performed before reporting the finalized value to the requested contract.

Validation layer is the mechanical part of the architecture. It is automatic and hence the validation client can be run by stakers with virtually no manual actions required.

1.4.2 Governance layer

It is desirable to have an oracle platform with fast response time. Hence subjective decisions have been separated from validation layer into a separate layer called “governance layer”. Subjective decisions requiring manual decision making are done in governance layer.

The goal of the governance layer is to make decisions to maximize stakers profitability, utility of oracle platform, flexibility to modify data feeds and collections according to changing environment and protection of stakers from various attacks.

1.4.3 Application layer

Application layer is composed of any applications using the oracle. Razor, being a general purpose oracle platform, is permissionless. Hence, any smart contract application can pay the fees to use the oracle's service.

2 Architecture

The Razor network consists of 3 layers:

1. Validation layer
2. Governance layer
3. Application layer

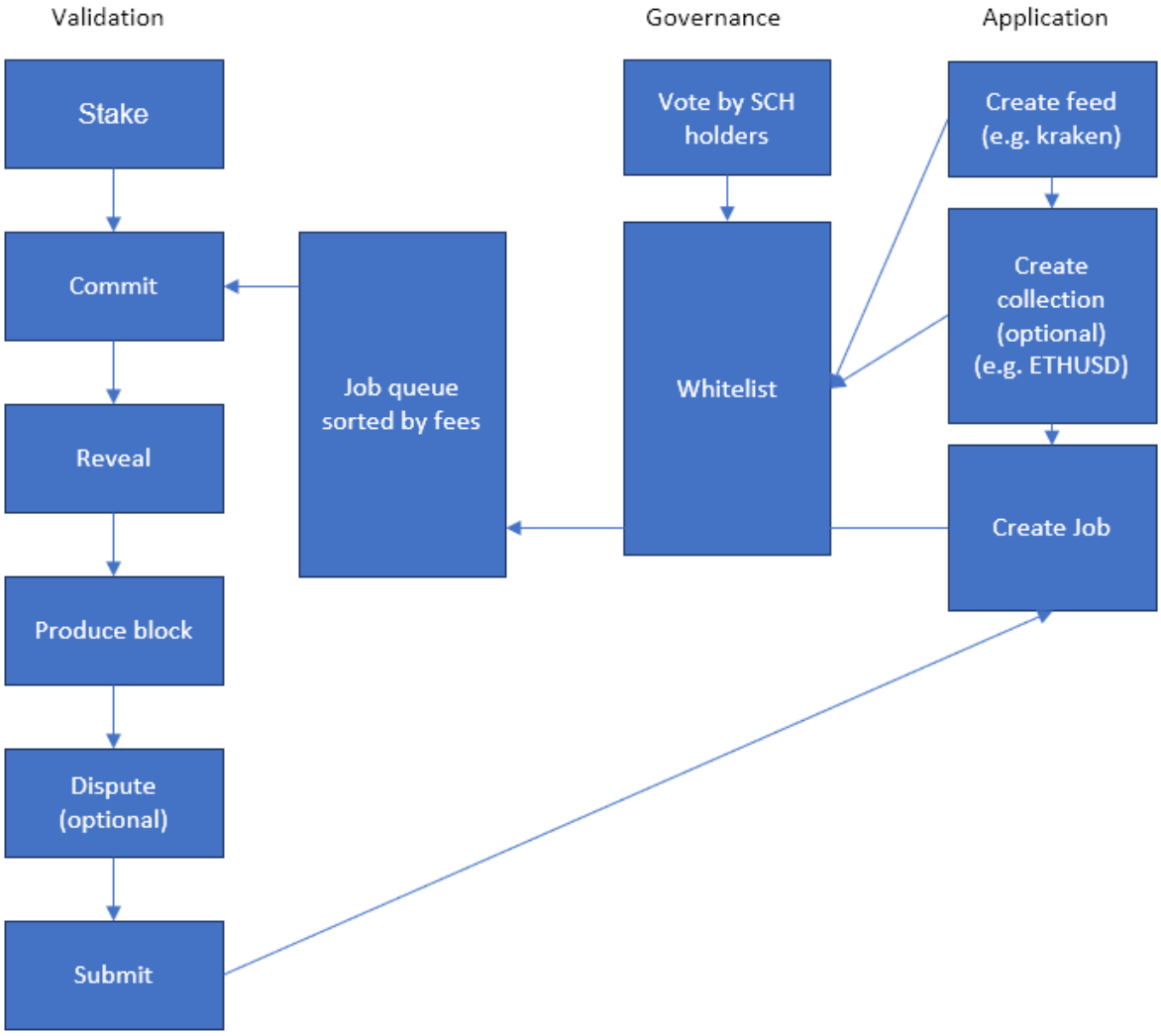

Figure 2: Process flow in Razor network

2.1 Schell - Native token or Razor network

Razor network will have a native ***utility*** token called "Schell" (abbreviated SCH, symbol Ş). Schells are ERC20 tokens on the Ethereum mainnet. These are necessary to perform a variety of activities in Razor network. There will be an initial minted supply of schells and the rest will be minted and distributed to stakers as block rewards.

2.1.1 Utility of Schell

Schells are necessary to perform following activities in Razor:

1. Staking
2. Paying fees for using the oracle
3. Creating and voting for proposals in Governance Layer

2.1.2 Supply Schedule

There can be a maximum of 1 billion schells in existence. The block rewards will be high in the beginning to encourage staker participation and will slowly decrease over time. More details about the supply schedule will be discussed in a separate paper.

2.2 Actors

1. Stakers
2. Clients

Stakers are the users who stake their schells to participate in processing jobs and reporting results to the network. In return they get rewarded through block rewards and fees paid by clients.

Clients are users who use the services of the platform to get values of various data points by paying the fees. The fees must be paid in schells.

2.3 Validation Layer

Schells can be locked in a smart contract by users called as “Stakers”. Schells must be staked on Razor platform to perform various actions and generate rewards. Stakers are rewarded to be honest and report values in consensus with the majority of stakers. The datapoint reported with majority consensus will be regarded as the “truth” adherent to the “Truth by consensus” approach.

We will also be measuring the consensus of the stakers along with other stakers and their availability in the form of Influence points. Influence points decide the amount of influence of the staker over the network.

Acting dishonestly may cause loss of influence points or stake.

Stakers in the Razor platform can perform the following tasks:

1. Process the selected jobs in the job queue by querying the mentioned datafeed/collection and processing it before reporting it to the oracle contract.
2. Reveal the secret and desired data-points
3. Propose a block if elected as a block proposer
4. Dispute blocks, if found invalid
5. Submit the results to the client smart contract, once finalized

2.3.1 Epoch

One cycle of the oracle is called an “epoch”. Each epoch is divided in 5 stages of equal periods. One stage consists of B blocks of Ethereum blockchain and hence, one epoch consists of 5B blocks. If we set B at 25, then, at the time of writing this paper, this would make duration of each epoch approximately 35 minutes.

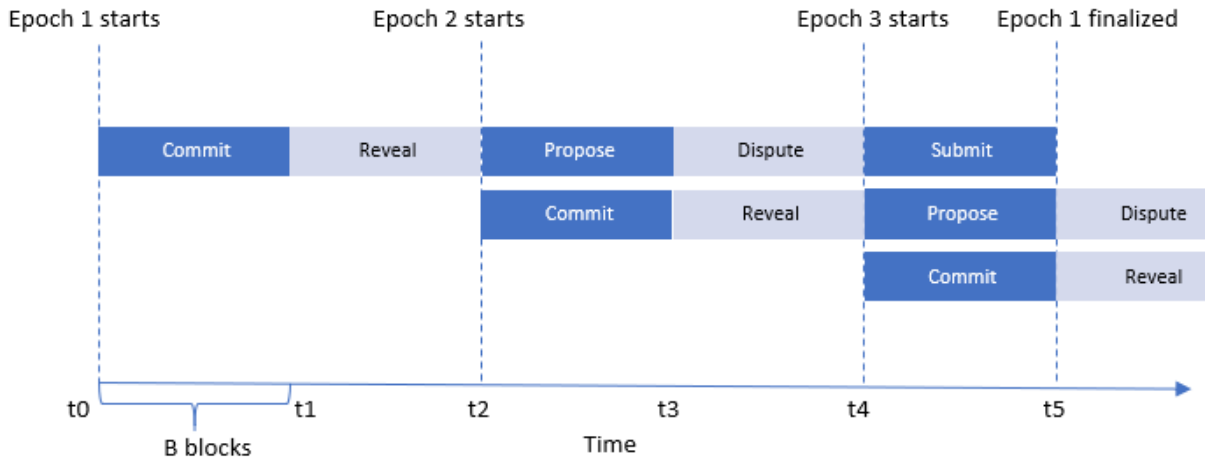

Figure 3: Epochs and their overlap

To make the protocol more efficient, there will be multiple epochs active at any point in time. New epoch starts every 2B blocks.

2.3.2 States

The Razor oracle has two States:

1. Commit
2. Reveal

Each state has a period of 25 blocks (approximately 7 minutes) of Ethereum mainnet and they alternate.

During ***Commit*** state, following actions can be performed: Stake, Commit results, Unstake, Withdraw, Propose block for epoch (e - 1), submit block for epoch (e - 2)

During ***Reveal*** state, following actions can be performed: Reveal results, Dispute block proposed in epoch (e - 1)

Here, e is the current epoch.

Do note that there can be upto 3 epochs running simultaneously in different stages, but they are in the same state as can be seen in Figure 2.

2.3.3 Job queue

Job queue consists of list of jobs that need to be processed by the oracle. The job queue is sorted by the amount of fees being paid. In every epoch, at most J jobs will be selected and processed by the stakers.

$$J = \frac{S_N \times L}{R}$$

Where,

S_N is the total number of active stakers

R is the Redundancy factor and determines how many stakers will report the value for each job

L is the Load factor, how many jobs with each staker process

The governance layer will be able to make changes to the value of R and L as necessary, which, in turn, decide the value of J.

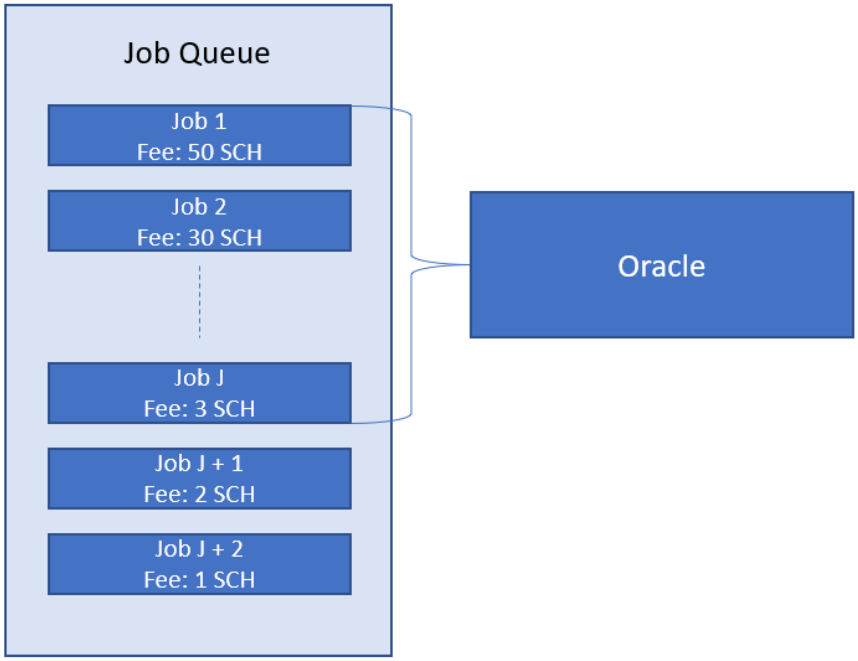

Figure 4: Selection of the jobs from the job queue

2.3.4 Actions

The following actions can be performed by stakers: Commit, Reveal, Propose, Submit, Unstake, Withdraw.

Following actions can be performed by anyone: Dispute, Submit Job, Stake

2.3.4.1 Stake

Staking involves locking ones "Schells" in the Razor oracle smart contract. Staking is required to process jobs and propose blocks in the Razor oracle platform. Users are incentivized to become a staker because they earn a fixed amount of newly minted schells called "block reward" in every epoch. They also earn the transaction fees paid by clients. However, staking also comes with a responsibility to keep the staking node active and behave honestly, or else penalties may be charged.

Staking can only be done during ***Commit*** state. A minimum of T_{min} schells must be staked to become a staker. If at any point in time, a staker's stake drops below T_{min}, she will not be able to participate in the network. Staker must pay T_{fee} tokens, which are burned, to join the network as a new staker. Stakers are subject to lock-in periods and will not be able to withdraw before it expires.

There is no upper limit on the maximum number of stakers. However, only a certain amount of stakers, $S_{winners}$ will be selected to participate in each epoch by a lottery. The chance of getting selected in each epoch is proportional to the influence of the staker.

A staker is selected to participate in the epoch if the following statement is true:

$$PRN < \frac{I_i}{I_m \times D}$$

Where,

PRN is the pseudo random number generated by each staker

I_i is the influence of Ith staker

I_m is the influence of the staker with most influence

D is the difficulty

Here, the difficulty D is adjusted each epoch so the selected number of participants $S_{winners}$ is equal to the desired value. If it is above the desired value, D will be reduced by 5% and vice versa. It is necessary to limit the number of active stakers each epoch in the network to avoid scalability issues, hence $S_{winners}$ should be set carefully by the governance layer to limit the maximum number of stakers in the network.

Only the stakers selected in this lottery will be able to participate in that epoch.

Stake action can also be performed by existing stakers to increase their stake by locking additional tokens.

In Ethereum, to discourage coordination of stakers through staking contracts (also known as staking pools), this action can only be called by an Ethereum account and not a smart contract.

2.3.4.2 Commit

If there are jobs pending in the job queue, stakers process them and submit the final data point. As Ethereum is a public blockchain, everyone can see everyone else's data points. This can cause various issues such as stakers piggybacking other stakers by copying their data points, trustless on-chain bribing attacks, influencing small staker's results by large stakers, etc.

Hence, we will be using a cryptographic commit-reveal scheme to keep the stakers' results secret. The stakers selected by lottery process described in **2.3.4.1** will be able to participate in this action. They must process **all** of the J jobs to be processed in this epoch, from the job queue and form a data structure called Merkle tree[6]. The stakers then combing the root of the Merkle tree with a secret salt before hashing it. This hash is the "commitment" of the staker to the results she arrived at.

Please note that the stakers process and commit all of the J jobs. But they will only reveal the values they are assigned to, in the reveal state. The jobs assigned to a staker are only revealed at the end of the commit phase, hence they must process and commit all of the J jobs honestly.

The stakers are heavily disincentivized to reveal their results by revealing their secret because if anyone reveals their secret in the commit state, the staker loses their entire stake.

Commit action can only be performed during ***Commit*** state. At the beginning of this state, J jobs from the job queue are selected based on fees. All the stakers must process these jobs. In case a staker doesn't perform this action, she will be penalized. Stakers must form a Merkle tree as shown below:

[6] Merkle tree is a tree in which every leaf node is labelled with the hash of a data block, and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes.

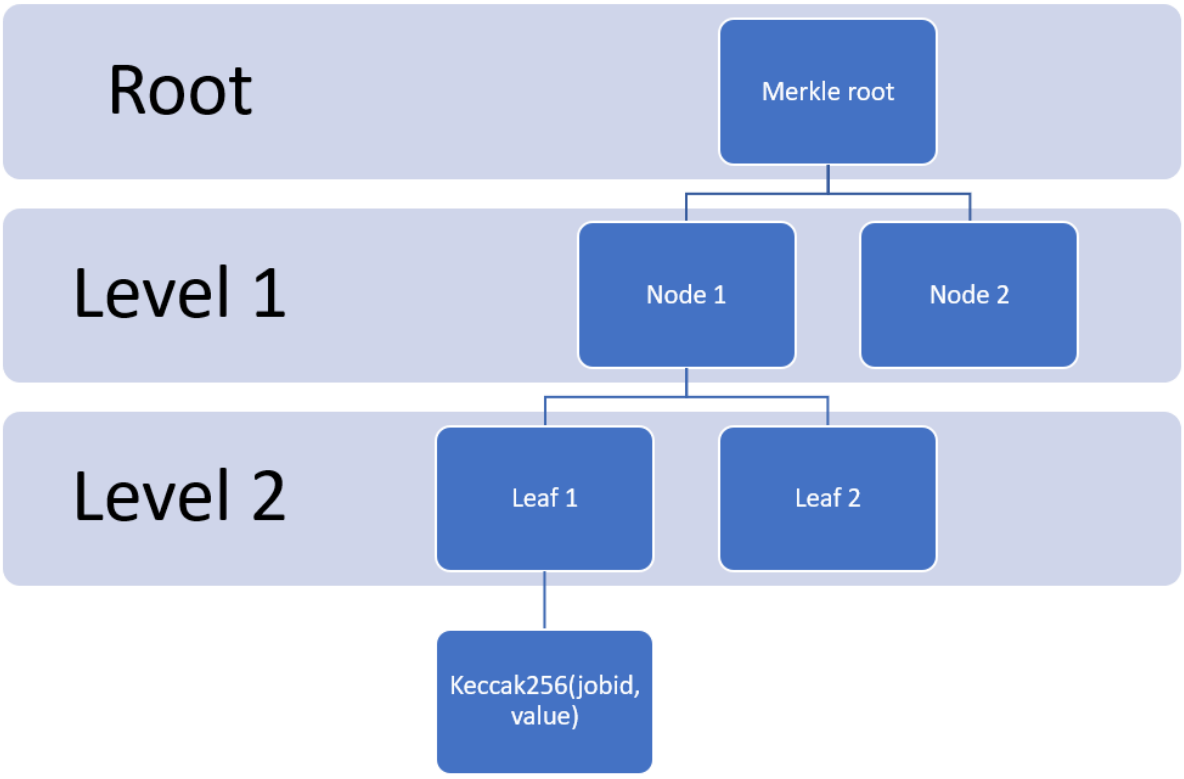

Figure 5: Merkle tree of commitments

A Merkle tree is a binary tree. Each of the nodes are labelled by the hash of its children and the leaves are labelled by the hashes of (jobId || data-point value)

In above figure we are assuming that we are processing 4 jobs (assuming J = 4). Every staker would need to process 4 jobs and would have arrived at 4 data-points.
Every staker must submit the following value to the oracle smart contract:

$$C = H(e \,||\, R \,||\, S)$$

Here,

C = Commitment

H = Collision Resistant cryptographic one way Hash Function. We will be using keccak256

e = epoch

R = Merkle Root

S = secret, a 32 bytes randomly generated salt

Stakers must locally generate and save this salt carefully, as it is required to reveal the results in reveal stage. Also, if the secret is stolen and revealed by someone else, harsh penalties will apply.

2.3.4.3 Reveal

This is the second stage of the reporting process. Stakers are supposed to reveal the secret they used in Commit stage as well as the results of the job assigned to them, along with the merkle proof proving that the submitted values are part of the commitment. This action can normally be performed in ***Reveal*** state. However, anyone can call this function to submit another staker's secret in ***Commit*** state to earn bounty and penalize the malicious staker.

Every staker will be assigned a job pseudorandomly as follows:

1. A pseudo random number will be generated using following salt:

$$PRN = PRNG(n \,||\, B_C \,||\, StakerId)$$

Where,

PRN = Pseudo Random Number
$PRNG$ = Pseudo random number generator, generates a number between 0 and 1
n = nonce
B_C = hash derived from Ethereum blocks during the commit state

2. The following equation will be evaluated to determine which jobs are assigned to the staker

n^{th} Job will be assigned to the staker if the following condition is satisfied:

$$\frac{n}{J} < PRN \leq \frac{n+1}{J}$$

Here,

n = job ordered n^{th} from the top of the list
PRN = Pseudo Random Number generated in earlier step
J = total number of jobs to be processed this epoch

E.g. Let's assume J = 4. A staker generates PRN and performs following comparisons to evaluate which job is assigned to her.

if 0 <= PRNG < 1/4, first job is assigned
if 1/4 <= PRNG < 1/2, second job is assigned, and so on.

Figure 6: Assignment of jobs to a staker

The above steps are repeated L times (where L is the load factor defined in 2.3.3) with incrementing nonces n = 1,2,3 …. L to determine which jobs are assigned to a staker. If staker does not reveal during reveal state, she will be penalized.

The staker must prove that she is only reporting the jobs as assigned to her, she is reporting all the jobs assigned to her and that she is reporting the committed values without changing them. The staker must also provide the leaf or node hashes to reconstruct the merkle tree.

As an example, let's assume J = 4 and jobs J_1 and J_4 are assigned to a staker. She must call the Commit action with following parameters:

$$Commit\ (e,\ S,\ J_1,\ R_1,\ J_4,\ R_4,\ L_2,\ L_3)$$

Where,

J_1 is the job ID of job 1
S is the secret used in commit state

R_1 is the result of job 1 as committed by the staker

L_2 is the hash of leaf 2

From Figure 5, you can see that this much information is sufficient to partially reconstruct the merkle tree and derive the merkle root. The merkle root and the secret will be used to reconstruct the commitment and it will be verified against the commitment made by the staker.

2.3.4.4 Propose Block

During propose state, any staker can propose a block, provided their current staked amount is above the minimum stake required and their influence is nonzero. A sorted list of stakers is pseudorandomly but deterministically calculable for each epoch. The probability of being higher up the list is directly proportional to the influence of each staker. The staker on top of this list gets the highest priority to propose a block. In case this staker does not propose a block, or proposes an invalid block, block from the second proposer on this list will be selected and so on. In case there are no valid blocks proposed, the epoch will end without a block and the jobs will be processed in the next epoch.

The following algorithm is used to prepare the block proposer priority list:

1. First we will select a staker pseudorandomly by virtually rolling a N sided fair die. This can be calculated programmatically as:

$$S_i = \lfloor PRNG(n \mid\mid B_R) * N \rfloor$$

Where,

S_i = Staker ID

$PRNG$ = Pseudo Random Number Generator function which utilizes provided salt

n = nonce

B_R = blockhashes[7] of reveal state blocks of current epoch

N = Number of stakers

2. Then we will evaluate the following equation:

$$\frac{I}{I_M} \leq PRNG(n \mid\mid S_i \mid\mid B_R)$$

Where,

I = Influence of the staker

I_M = Influence of the staker with highest influence

The above steps are repeated with increasing nonce ($n = 1,2,3,4,...$) and whenever the second statement is evaluated to be true, that staker S_i is added to the end of the block proposers list. Stakers who are already in the list are skipped.

[7] Each block in blockchains such as Ethereum has a hash. This hash is virtually random and depends on the contents of the block and hash of previous block. Here blockhashes of all the blocks during the reveal state will be combined to increase entropy of the salt.

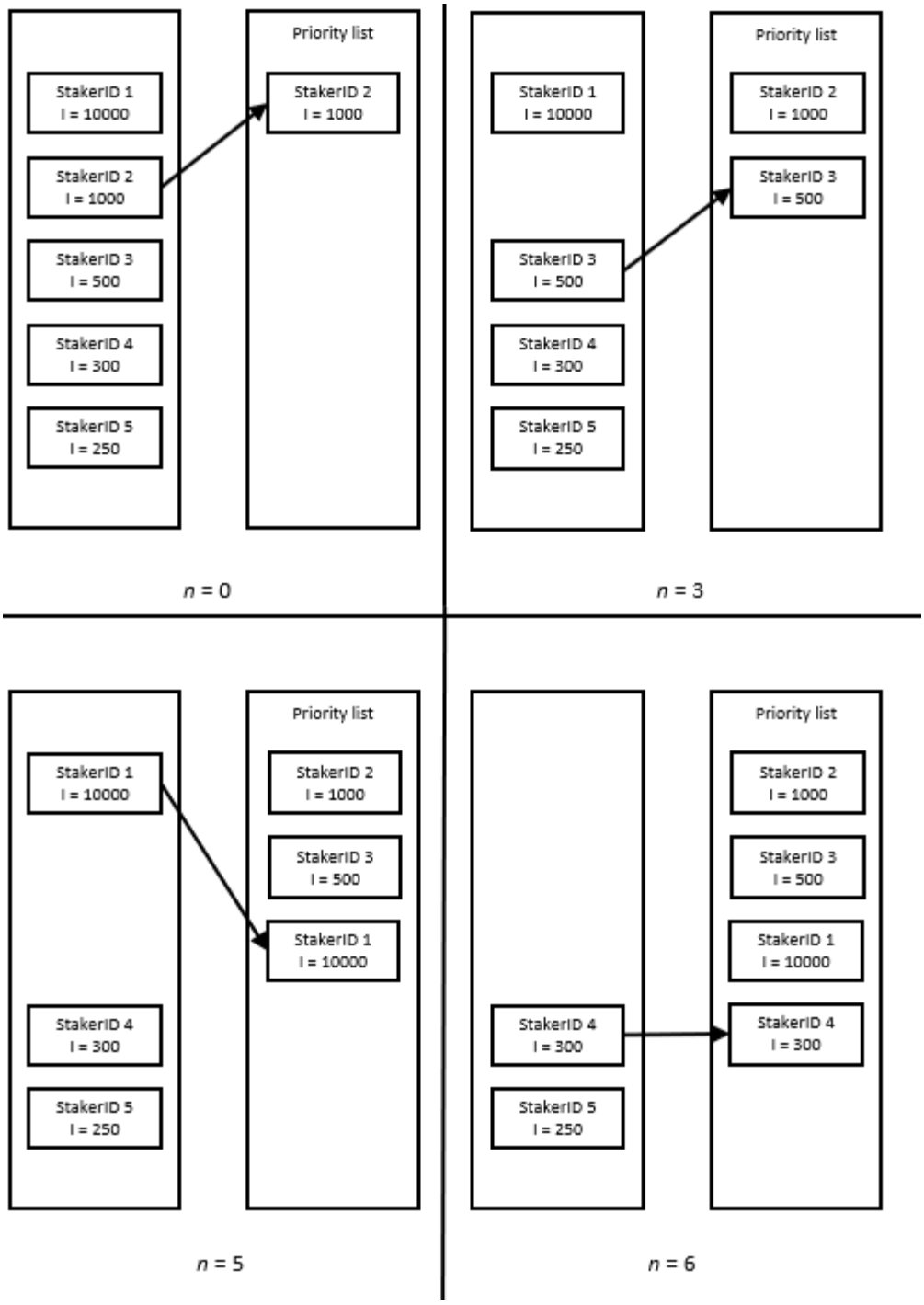

Figure 7: Selection for the block proposer list

To propose, the staker must call the following function in the Razor validation smart contract:

$$Propose(e,\ n,\ S_{MI},\ M_1,\ M_2,\ M_3, ...M_J)$$

Here,

e = current epoch number
n = nonce
M_J = Median for job J
S_{MI} = Staker ID of the staker with maximum influence

A valid proposal with lowest nonce and S_{MI} with highest influence will be selected as a block. Even though the value of S_{MI} is available in the smart contract, calculating it will require iterating through all the stakers. Hence to overcome the technical challenge, it is instead proposed by the stakers.

Since all the values to be submitted by the stakers are deterministic from the data available inside the blockchain, there should be no reason for miscalculations and deviation of the values from the true values. Hence harsh penalties will be applied and stakers can lose their entire stake if an invalid block is proposed.

If a block is proven invalid during dispute state, the next block in the queue will be selected as a candidate block to be finalized. And the process can repeat. If no blocks are remaining in the

2.3.4.5 Dispute Block

If an invalid block is found, anyone can dispute it by performing on-chain aggregation calculation of the disputed job. If the calculation is proven to be incorrect, the next block in the priority list will be chosen as the valid block. The process repeats till valid block is found or time runs out.

If a block is proven invalid, 100% of the stake is slashed. 50% of it is burned and the other 50% is rewarded as a bounty to the disputer.

2.3.4.6 Unstake

If staker wants to withdraw stake, she must call Unstake function. This function can only be called in commit state. Once called, she has to continue being an active staker for 1 dynasty (1000 epochs), before finally being able to perform the withdraw action.

2.3.4.7 Withdraw

Once unstake is called and lock period is completed, staker can withdraw the stake during commit state. Once the lock-in period is complete, staker will not be able to actively participate in reporting and other functions. Withdrawing the stake will reset the stakers reputation points to zero.

2.3.4.8 Submit results

This action submits a finalized result to the requesting contract. This can be done during Commit state. The elected block proposer must perform this action for all of the J jobs for this epoch, or else she will not earn the block reward.

2.3.4.9 Submit job

Anyone can submit a job to the job queue provided it is whitelisted by the governance layer. If the job is not whitelisted, it must be submitted to the governance layer so that the stakers can approve or reject it.

2.4 Incentivization

It is necessary to design a balanced incentivisation system. If the incentives are not substantial enough or if the penalties are too harsh, the platform will not attract a large number of stakers.

2.4.1 Block reward

A block reward of B schells will be awarded to the stakers, if the following is true:

1. Staker proposes a valid block in time
2. Staker has the highest priority of becoming block producer for the current epoch
3. No one successfully proves the block as invalid during dispute period
4. Staker submits the block to the client contract

2.4.2 Fees

The fees paid to process the jobs by the clients are distributed to the stakers who process them. The fees must be paid in Schell tokens. The fees are distributed in proportion to the influence of the participating stakers.

2.4.3 Influence, Reputation and Maturity

Reputation is a measure of staker's consensus with other stakers and availability in Razor network. Whenever a staker performs stake action, she gets a small amount of reputation points. Reputation points increase or decrease based on whether the staker's votes are in consensus with others.

Reputation of a staker at an epoch can be calculated from the following formula:

$$R_i = Minimum(Log(m + s), R_C)$$

Where,

R_i is the reputation of i^{th} staker

Log is the natural logarithmic function

m is the maturity of the stake. Initially it the amount of epochs the stake has been locked. However, this value will be reduced if a penalty is applied

s is the smoothing factor, to reduce the high rate of growth at an early stage of the curve

R_C is the upper limit on reputation

This logarithmic curve was chosen so that the reputation grows slowly over time. This prevents hit-and-run kind of attacks where an attacker stakes, reports maliciously, leaves and immediately joins the network again as a new staker to repeat the attack.

The following equation will be used to calculate penalty for stakers who report values outside consensus. This penalty will be applied on the maturity of the stake.

$$Percentage\ Penalty\ on\ maturity = 100 \times \frac{(M-x)^2}{M^2}$$

Where,

x = the reported datapoint by a staker between 0 and 2*M

M = the weighted median calculated from all reported data points

For x > 2*M, penalty is 100%

Plot obtained from above equation is shown below, assuming M = 100.

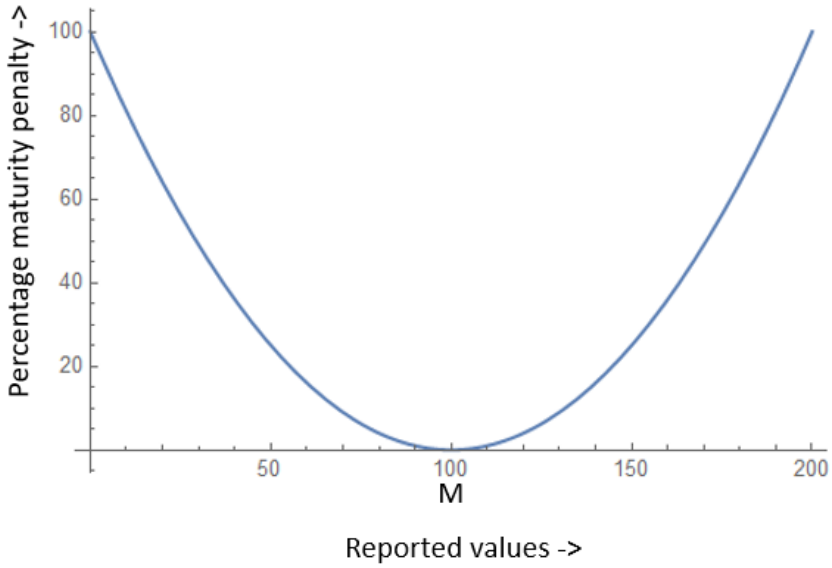

Figure 8: Maturity penalty

Reputation points along with stake of the staker determine the influence of the staker.

$$Influence = Reputation \times Stake$$
$$I_i = R_i \times T_i$$

Where,
I_i is the influence of i^{th} staker
R_i is the reputation of i^{th} staker
T_i is the number of tokens staked by i^{th} staker

Influence points determine a staker's influence over the network. Influence points are used as "weights" when aggregating results from all stakers for a job. Also higher influence increases chances of becoming a block producer.

2.4.4 Penalties for misbehaviour

If a staker proposes an invalid block, it can be proven by anyone by performing the aggregation calculations on the blockchain. Since all the data for creating a valid block is available on chain, and everyone is assumed to be using the same client without improper modifications, there is no non-malicious reason to propose an invalid block. Hence, entire stake of the malicious staker will be slashed. Half of it will be burnt and the other half would be rewarded to the disputer.

For each epoch a staker does not commit a result, she will get 1% of her influence. While committing but not revealing data points in an epoch will result in a penalty of 5% of her influence

2.5 Security

Razor uses widely used cryptographic primitives, which are proven to be secure and well optimized. keccak256 hash function, used for commit-reveal scheme and for generating seed from blockhashes for random number generator, is collision resistant.

2.5.1 Economic security

Incentives are carefully designed to reward honest behaviour and punish malicious behaviour. To perform a takeover attack, 51% of influence needs to be controlled by one or several entities colluding together. However a takeover attack makes the network unusable and can have devastating consequences on the value of Schells in the market. Hence the attackers, controlling 51% of influence (and hence, likely a comparable amount of Schells) are heavily disincentivized to perform takeover attack or to maliciously influence the values reported by the oracle.

However, if the money earned by performing the attack is more than the cost to perform the attack, the attack can be profitable. Hence assuming the worst case scenario, the sum of market caps of the applications dependent on Razor oracle should be 50% of the stake. This is the economic security provided by the network.

Please do note that the above case assumes worst case scenario in which stakers are able to freely coordinate with each other and completely trust each other. However due to inefficiencies of the real world and due to anti bribing and anti collusion design used in the protocol, in reality a much larger economy can be secured by the Razor protocol

2.5.2 Attacks

Being a decentralized and open protocol, Razor network must be resilient to every possible attack. It is important for the oracle to provide high economic security guarantees, otherwise it's not feasible to build building large scale financial applications by utilizing its service.

2.5.2.1 Influence of large stakers

Razor oracle consists of focal point game where actors report the true value T because they feel all other actors will report T because it is the true value and it is not feasible to trustlessly coordinate with other stakets and decide any other value. Hence is important that all actors are voting independently without coordinating with each other and without influencing each others votes.

An example of this attack is where an attacker, who has a large stake, reports value A. Let us assume this value has large difference with true value T. Other stakers can see this value on the blockchain as it is a transparent protocol. It would be in their interest to report the value A rather than T, because they see a very large percent stake voting for A and the weighted median will likely be moved closer to A rather than T.

The effect may not necessary be malicious. Some stakers may choose to piggyback on other stakers to save their own resources. They can just copy their fellows stakers votes. This reduces economic security of protocol.

To address these issues, Razor oracle uses commit-reveal scheme. The stakers' votes are secret but their commitment is recorded on the blockchain using a cryptographic hashing function. The staker's only reveal their votes during reveal state when it is not possible to commit anymore. If the staker publishes their secret before reveal phase, anyone can reveal this secret to earn a bounty and slash that staker's stake and influence.

2.5.2.2 Takeover

As discussed in 2.5.1.

2.5.2.3 Bribing

A bribing attack is where the attacker bribes the stakers to perform actions to her favour. For the oracle to be bribe resistant, the following must be true:

$$\textit{Profit from bribing} < \textit{Cost of Bribe}$$

Razor network aims to provide a high degree of economic security. Since it is impossible to know which stakers will be assigned to attacker's job in the commit state, the attacker will need to bribe 51% of the network.

In addition to that, due to commit-reveal scheme used for reporting and harsh penalties applied for revealing secret prematurely, trustless bribing attacks are not possible.

2.5.2.4 Collusion

It is possible that stakers may collude and fix the results of the jobs. The colluding group must have a high enough stake otherwise their attempt will fail as their values will not be in consensus with values reported by other stakers. Hence to be effective, the colluding group must have 51% of influence over the network.

2.5.2.5 Griefing

A griefing attack is defined as an attack where an attacker causes inconvenience or loss to others while not making any profit for herself.
Various kinds of potential griefing attacks are:

1. Not commiting results
2. Committing and not revealing results
3. Revealing random or false results
4. Not proposing block
5. Proposing invalid block
6. Voting in governance layer in an irrational manner

The incentives and penalties of the protocol are carefully designed to penalize such behaviour. Any values reported which are against the consensus will attract Influence penalties and make such attacks unsustainable.

3 Governance

World is continuously changing. Oracles, being dependent on external data, must be flexible enough to adapt to the changing environment. External websites may become defunct, compromised, or behave in a byzantine manner. Since staking clients in Razor network are automated, validators cannot be expected to react to such changes immediately.

Due to these reasons, we will be separating manual, more subjective form of decision making into an on chain Governance layer, while the automated objective decision making part stays in validation layer.

Governance layer performs the following functions:

1. Enable the oracle network to do subjective decision making on acceptance or rejection of data sources
2. Accept, reject or modify data source collections
3. Change parameters of validation layer

Whitelisting and blacklisting datasources is an important function of the governance layer. It protects the stakers from selective misinformation and Denial of Service attacks.

3.1 Data sources

Data sources contain metadata about jobs. A datasource contains following fields:

1. Datasource ID
2. URL
3. XHTML / JSON / Regex selector
4. Symbol (optional)
5. Pair (Optional)
6. Frequency with which the data is updated (optional)
7. Availability (time periods when it is available, optional)

3.2 Collections

These are curated collections of data sources to provide a specific datapoint from various sources.

E.g. ETH/USD. Stakeholders of the governance layer make decisions on creation, modification and deletion of collections.

Collections can have custom combination functions to combine data from various data sources. E.g. ETH/USD collection can have an ETH/EUR source and an USD/EUR datasource. These data sources can be used to derive an ETH/USD collection.

3.3 Criteria for whitelisting datasources

The jobs can only use data sources and collections whitelisted by the governance layer. If a datasource or collection is not whitelisted, anyone can create a request to whitelist them. The stakers can decide to approve or reject data sources at their will.

The primary advantage of whitelisting is to protect stakers from malicious requests. Hence it is in their own interests to carefully review the requests before approving.

The following can be regarded as guiding principles when making the decision.

The data source should:

1. Be reputable and well known
2. Should handle heavy load
3. Should respond reasonably fast

4. Should not respond or behave in a byzantine manner
5. Reponses should not be too big (e.g. < 1 MB)
6. Should be freely accessible and should not require a login, proxy client, etc.

3.4 Criteria for adding data sources to a collection

Collections are curated collections of datasources. Collections can be used to provide long term data feeds without relying on any single centralized datasource. It is the duty of the stakers to curate collections and keep them up to date to increase the utility of the network. To add a data source to a collection, it:

1. Should not be legally, morally or technically “compromised”
2. Should preferably handle real asset versus derived asset (e.g. USD not USDT)
3. Reported values should not be outliers in any scenario
4. Should have sufficient liquidity, if it is an exchange
5. Should be confirmable from multiple independent data sources

3.5 Voting

Voting can be done by stakers in the network. The influence of the staker determines the weight of the of the vote. Even if someone does not want to actively participate in other staker actions and only wants to vote on governance issues, can stake without any problems. This is because their funds will be safe as long as they don't propose an invalid block.

3.6 Decisions on parameters of validation layer

Every dynasty (every 1000th epoch) of the validation layer is a governance epoch, where stakers can vote and decide on various parameters of validation layer.

4 Simulations

4.1 Influence

Simulations were performed with 20 stakers for 10000 epochs. The stakers had the following amount of tokens as stake:

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40]

It was assumed that the stakers were voting on the true value of 50 with a randomly generated deviation as per a binomial distribution.
The probability density for the binomial distribution is:

$$P(N) = \binom{n}{N} p^N (1-p)^{n-N},$$

where, n is the number of trials, p is the probability of success, and N is the number of successes. Here we used $p = 0.5$, $n = 1000$ and divided the obtained values by 10 to achieve the desired distribution.

We assume there is a 1% chance per epoch for each staker that they will defect and report a very large value (> 2*M, where M is the weighted median).

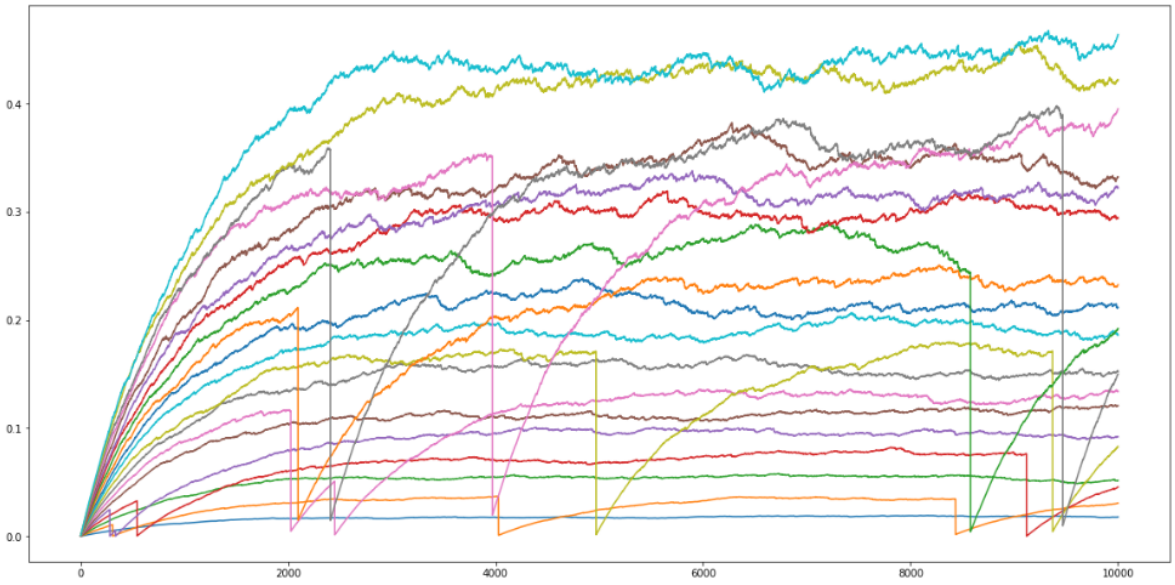

Figure 9: Influence of stakers over epochs

As can be seen from the chart above, the influence of the staker drops by a large amount when they defect, and then slowly rises according to a logarithmic curve. This makes repeated attacks less frequent and unlikely.

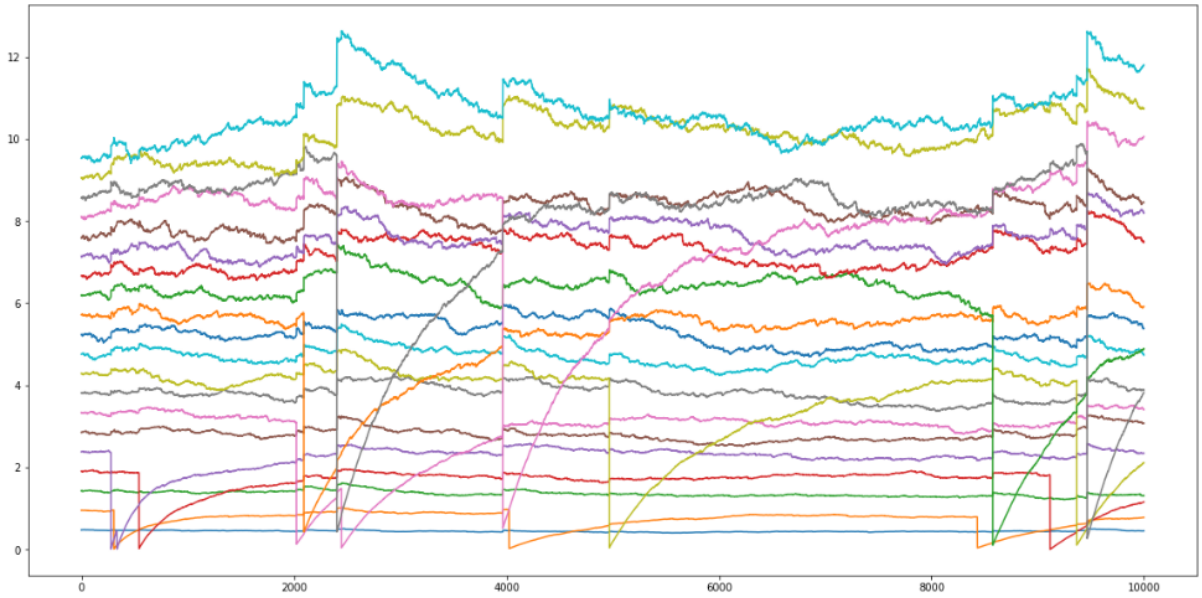

Figure 10: Percentage influence of each staker per epoch

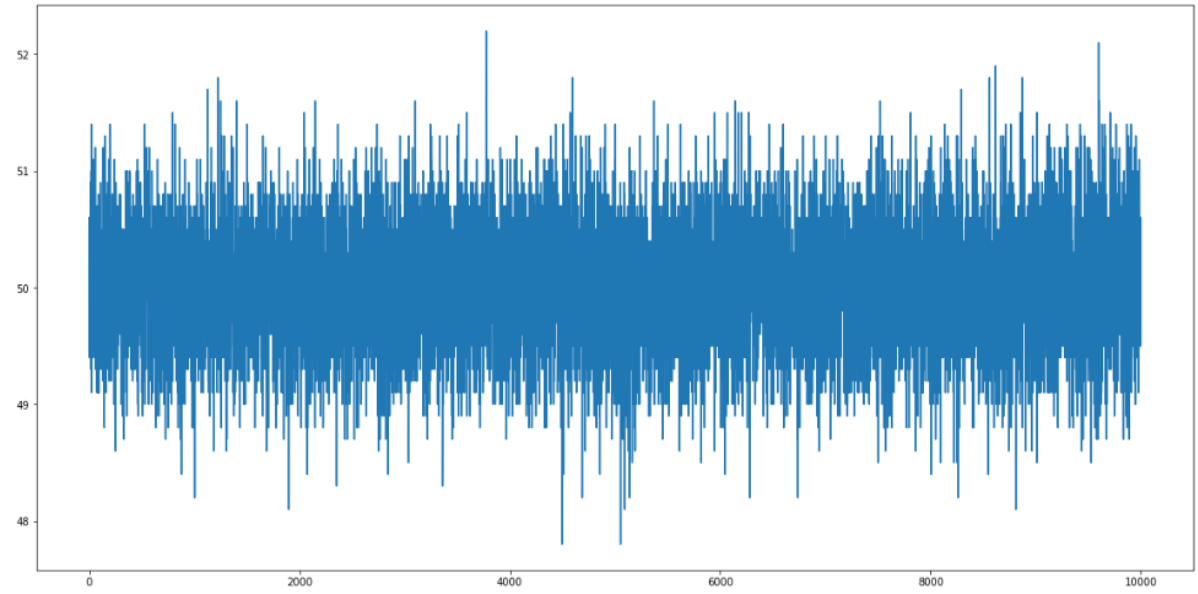

Figure 11: the final result obtained per epoch

As can be seen above, the protocol is resilient towards attacks and byazntine behaviour as long as a single entity (or colluding entities) do not control 51% of the influence.

4.2 Staker selection by lottery

Simulations were run to test the difficulty adjustment and staker selection algorithm. As can be seen in the chart below, the algorithm works as intended and maintains the number of stakers at around the desired value of 1000. Difficulty displayed is scaled by a factor of 1000.

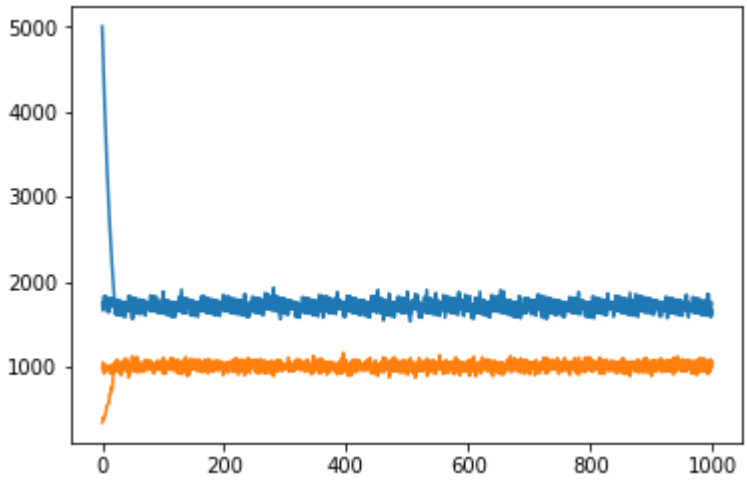

Figure 12: Difficulty adjustment and staker selection algorithm

5 Scalability

Due to the design of the platform, it becomes necessary to perform onchain multiple transactions by each staker for each epoch. This can be quite expensive especially for small stakers as the rewards gained by staking may not cover the transaction costs.

To solve this problem multiple scalability solutions are being researched. We will explore a few of the ways to solve this problem.

5.1 Trustless reporting pools using BLS signature aggregation

Reporting pools could be designed for aggregating commitments and revelations. Care must be while designing such an architecture to prevent manipulation by the pool operators.

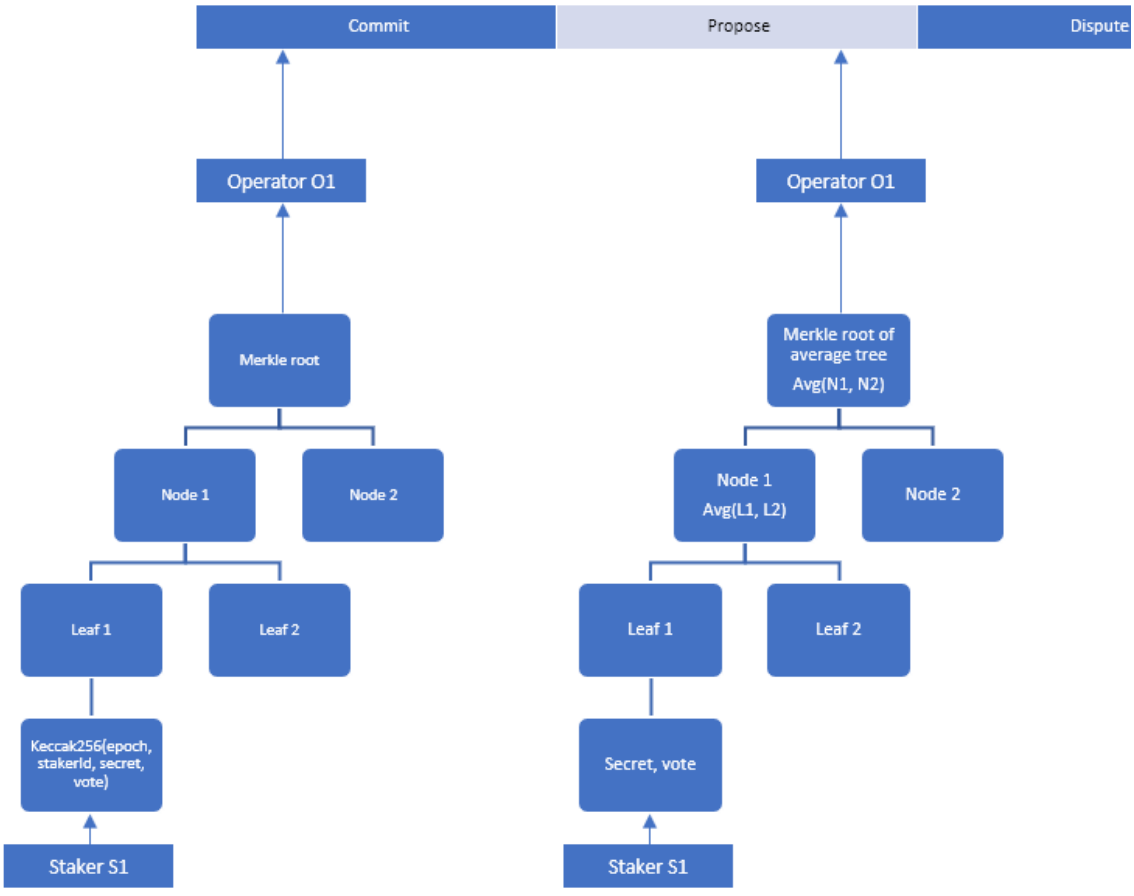

Figure. 13 Architecture of a Trustless reporting pool

5.1.1 Staking

1. Anyone can be a pool operator by staking a large amount of schells.
2. Stakers stake onchain as usual. Stake needs to be equal to a fixed number of schells (e.g.32). If staker wishes to stake more, she needs to create another staking account.
3. Pool operators operate a centralized web server where they collect input from stakers and publicly publish the data for transparency purpose.
4. The operator can choose to charge a percentage of earned rewards as fees.

5. Stakers should join the pool via a smart contract transaction, thereby delegating the reporting rights to the operator. They can choose any operator according to their track record, degree of transparency, fees charged, etc.

5.1.2 Commit

1. Operators collect commitments during the commit phase and create a merkle tree of commitments.
2. The merkle tree is distributed to all stakers in the pool, who should sign the merkle root only on validating and verifying the data.
3. The signatures will be aggregated using a m-of-n BLS threshold signature scheme. ⅔ of the stakers must sign the merkle root, for it to be valid.
4. In case the merkle root in invalid, the stakers must exit the pool onchain.
5. If more than ⅓ of the pool exits in the same epoch, the commitment made by the pool becomes invalid.
6. Operator records the merkle root of all commitments on the Razor smart contract.
7. Operator also calculates reputation of each staker, merklizes the data, gets the merkle root signed off by the stakers and publishes the merkle root onchain in this state.
8. In case of inactivity or withholding attack by the staker, the pool operator can kick them out.

5.1.3 Reveal

1. The secrets are gathered in a merkle tree and the root of it is signed off using BLS signatures of the stakers, similar to the commit phase.
2. The merkle tree is a modified merkle sumtree. We will call it merkle average-tree, with each node of the tree containing the average of value present in its children.
3. Since all the stakes are of the same value, the average of all votes is the same as weighted median of all votes, hence the value present in the merkle root is the weighted median of the votes present in the tree.
4. During reveal phase, the operator publishes the merkle root of secrets with weighted median value onchain along with the BLS threshold signature.
5. Similar to commit state, ⅔ of the stakers must sign off on this root or else it fails.
6. If stakers do not agree with the root, they must exit immediately.
7. If ⅓ of the pool exits in the same epoch, the pool is omitted from the current epoch.

5.1.4 Dispute

1. In the dispute phase, anyone (we will call them bounty hunters) can challenge an operator to provide merkle proofs of any leaf for any of the merkle roots published by them, by paying a small fee.
2. Since the revelation tree is a merkle average tree, the individual nodes of the merkle path can help the bounty hunter to find anomalies and ask for proof of specific leaves.
3. The operator or anyone with the knowledge of the proofs can respond on chain.
4. If no one provides the proof, the pool (along with all the stakers currently in the pool) will be slashed.
5. If the proof is successfully provided, the fee is earned by the pool and the next challenge will require twice the amount of fees.
6. Any stakers of the pool who exited the pool due to malicious behaviour by the pool operator can provide proof of such behaviour since they may have access to it when

they were part of the pool. The operator along with the pool will be slashed and the bounty hunter will be rewarded.

5.2 Deploying on a Layer 2 platform

The network can be deployed on a third party scalability network such as plasma, Matic network, Connext network, counterfactual state channels or any other ethereum sidechain solution.

6 Applications

Any application which depends on real world data can utilize Razor network to provide data points in a decentralized and trustless manner. Razor network is especially designed for long term decentralized applications requiring a high degree of decentralization and economic security. Decentralized finance applications are especially suitable since they almost always require such a datasource.

We will explore one example of such applications and explore how it can use Razor network. More applications are listed in 1.1.

6.1 Synthetic assets platform

We will explore how one can develop a Synthetic assets platform utilizing Razor network as an oracle service provider. A synthetic assets platform (Also known as a Delta one platform) provide a way to speculate on the value of any asset without actually trading that asset. A decentralized data source is a crucial component of such an application as the security and utility of the entire application depends on the datafeed.

A synthetic assets platform can be built using Razor network in following way:

1. The application developer can propose various data feeds and collections, as required by the application, to the governance layer.
2. The governance layer approves the data feeds and collections as long as they are valid and follow certain guidelines.
3. Users can provide collateral to mint new assets according to data-feed values. collateral can be SCH, ETH, etc.
4. Users can burn assets anytime according to price-feed values to get back their collateral.
5. As an example of an asset that can be created using the application, consider sUSD, a stablecoin pegged to the value of USD. Ether can be used as a collateral and ETH/USD price-feed can be served through Razor network as a reference for the minting/burning process.
6. When assets are requested to be minted/burned the next future available price-point will be taken as reference.
    a. E.g. if Alice requests to mint sUSD at 10am, the last traded price at the beginning of the next epoch will be used as reference.
7. When a position is under collateralized, anyone can liquidate a position by creating an update job for the oracle.
8. To long, buy a synthetic asset off the market. To short, mint it and sell it on the market.

7 Future work

7.1 Scalability improvements

Current architecture is based entirely on Ethereum mainnet. This is to provide a high degree of decentralization and censorship resistance from day one. We will continue to do research on scalability to develop a layer-2 scalability architecture which can provide similar guarantees.

7.2 Improvements to the governance layer

Onchain governance is an ongoing area of research. Improvements will be made to the governance layer over time according to the latest research.