Razor network: A decentralized oracle platform

Hrishikesh Huilgolkar
CTO, Razor network
Revision: 11th June 2019
Early draft for feedback. V 0.1

Abstract

Decentralized technologies are revolutionizing finance. Applications such as decentralized lending, stable currencies, prediction markets and synthetic assets are being researched and built on top of decentralized platforms. 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 are vulnerable to certain attacks due to 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 the 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 decentralized applications.

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 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

1 Introduction

1.1 Motivation

Decentralized networks, through the use of smart contracts, are disrupting finance by removing need of intermediaries and opening up equal access to everyone. Applications which were previously unimaginable are now possible. 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 have access to the outside world.

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 a 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[1], gnosis[2], MakerDao[3] and general purpose decentralized oracle platforms Truthcoin[4], SchellingCoin[5] Chainlink[6] and Witnet[7]. The current work is inspired by SchellingCoin and Truthcoin protocols.

Developing a decentralized oracle is deemed a challenging problem. This is due to possibility of multiple kinds of attacks such as collusion, takeover, griefing and so on, 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 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 proposed 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 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 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 very small 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 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 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.

In Razor platform, mechanisms and incentivizations are carefully designed to provide high economic security guarantees so application developers can easily integrate it in their application.

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. We propose a governance layer to protect the stakers from such attacks.

1.3 Design Goals

Design decisions have been made with following goals in mind:

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

Due to the high economic security, decentralization and censorship resistance properties of the Ethereum blockchain, we have decided to use it as the underlying platform. In future we will explore utilizing a Layer-2 scalability solution if similar guarantees are possible.

We have kept long term viability in mind when designing the protocol. Smart contract applications need a oracle solution that can be depended on over long term.

1.4 Architectural overview

Razor network consists of 3 layers

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

[5] Here, economic security is defined as the amount of money required to compromise the network.

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

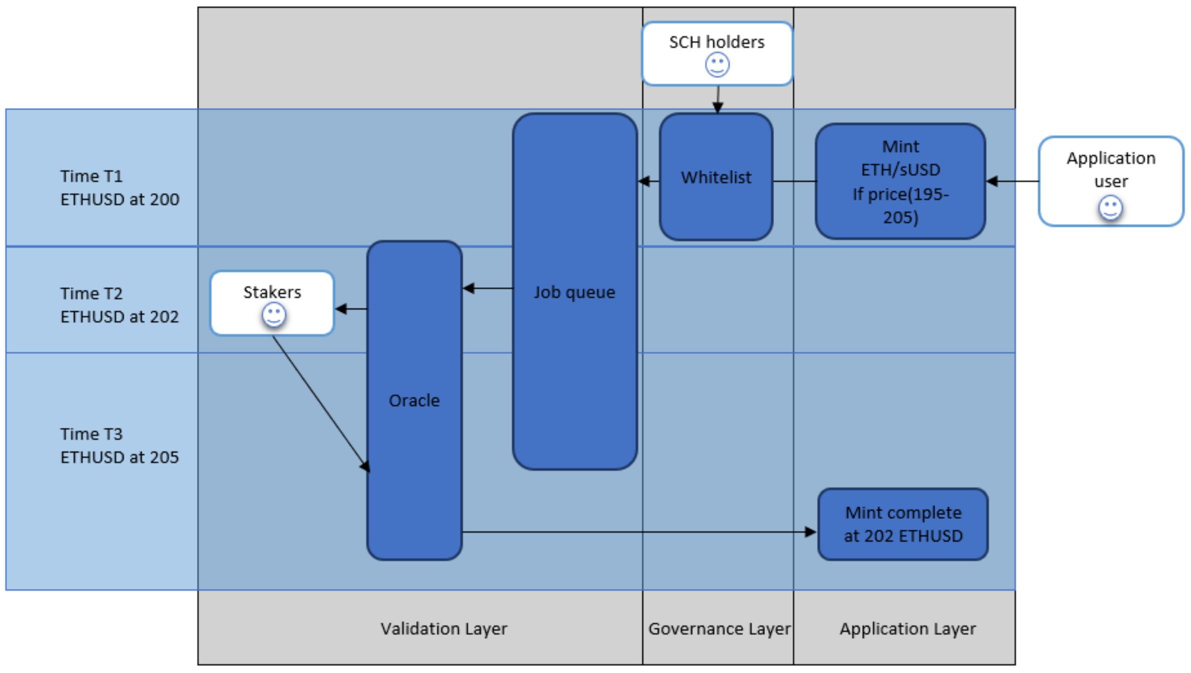

Figure 1: Architectural overview

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 more 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

2.1 Native utility token

Razor network will have a native token called “Schell” (abbreviated SCH, symbol Ş). Schells are ERC20 tokens which are necessary to do a variety of activities in Razor network. There will be an initial minted supply of shells and the rest will be minted and distributed to stakers as block rewards.

2.2 Inflation schedule

New tokens will be minted according to following schedule.inflation will be high initially then slowly taper off [todo]

2.3 Actors

1. Stakers
2. Clients

2.4 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 get rewards. Stakers are rewarded to be honest and report values in consensus with the majority of stakers. The datapoint reported with majority consensu will be regarded as the “truth” adherent to the “Truth by consensus” approach.

We will also be measuring the honesty and track record of the stakers 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 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

2.4.1 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.

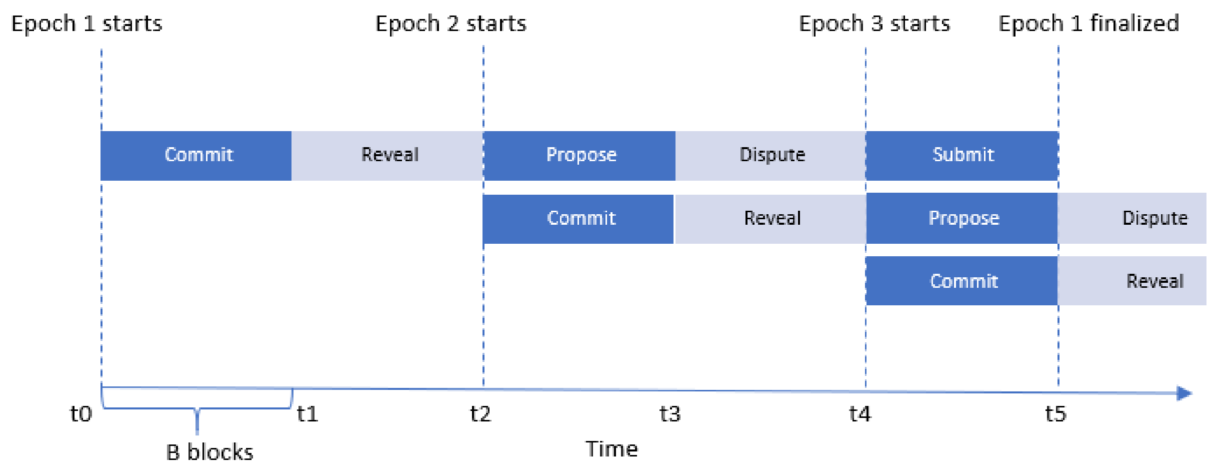

Figure 2: Epochs and their overlap

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

2.4.2 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.

2.4.3 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.

$$J = \frac{S}{R}$$

Where,

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

The governance layer will be able to make changes to the value of R as necessary.

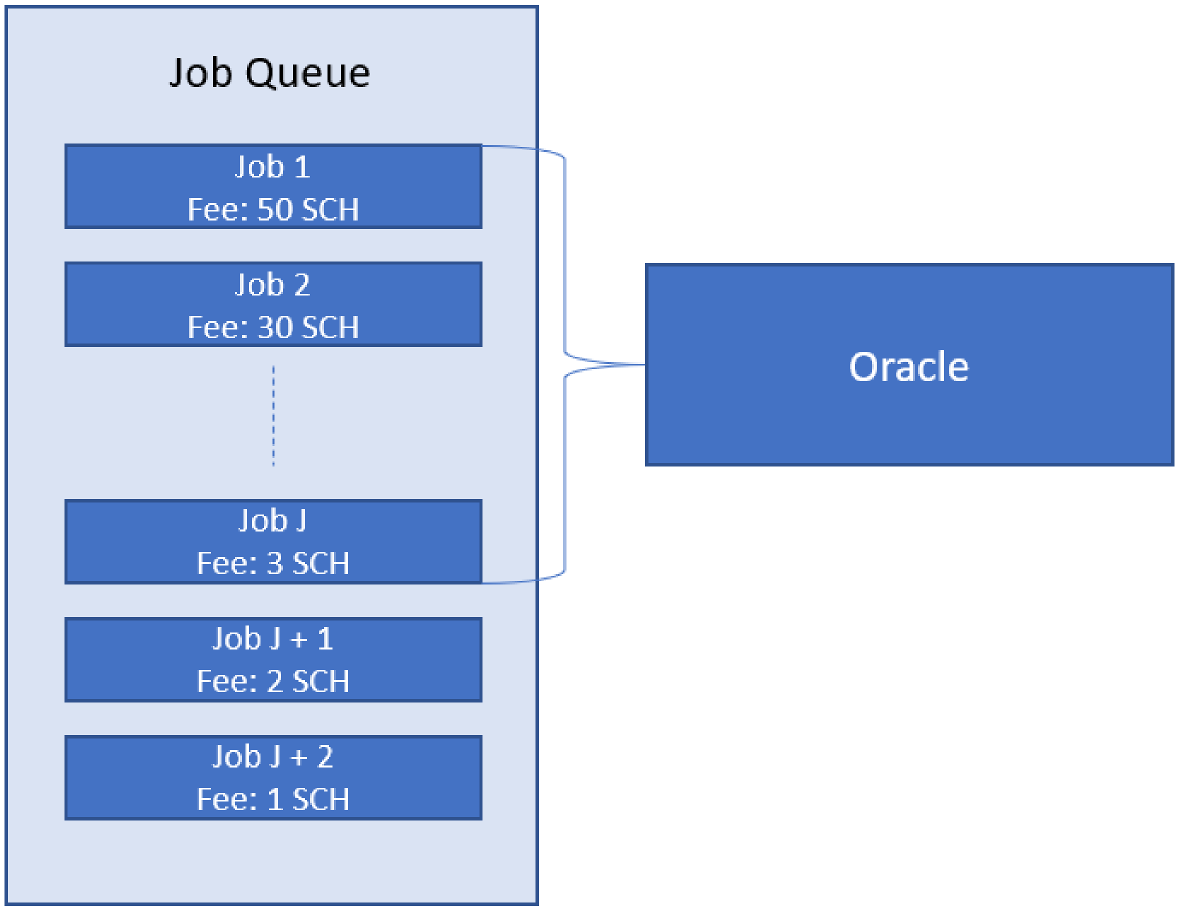

Figure 3: Selection of the jobs from the job queue

2.4.4 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

2.4.4.1 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, it also comes with a responsibility to keep the staking node active and behave honestly, or else heavy 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. It is necessary to limit number of active stakers in the network to avoid scalability issues, hence this parameter will be set carefully to limit maximum number of stakers in the network. 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.

Stake action can also be performed by existing staker to increase her 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.

2.4.4.2 Commit

If there are jobs pending in the job queue, stakers process them and submit the final datapoint. 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' votes secret. The stakers process all the jobs in the queue and form a data structure called Merkle tree[7]. 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 he arrived at.

The stakers are heavily disincentivized to reveal their votes by revealing their secret. This is 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, top 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:

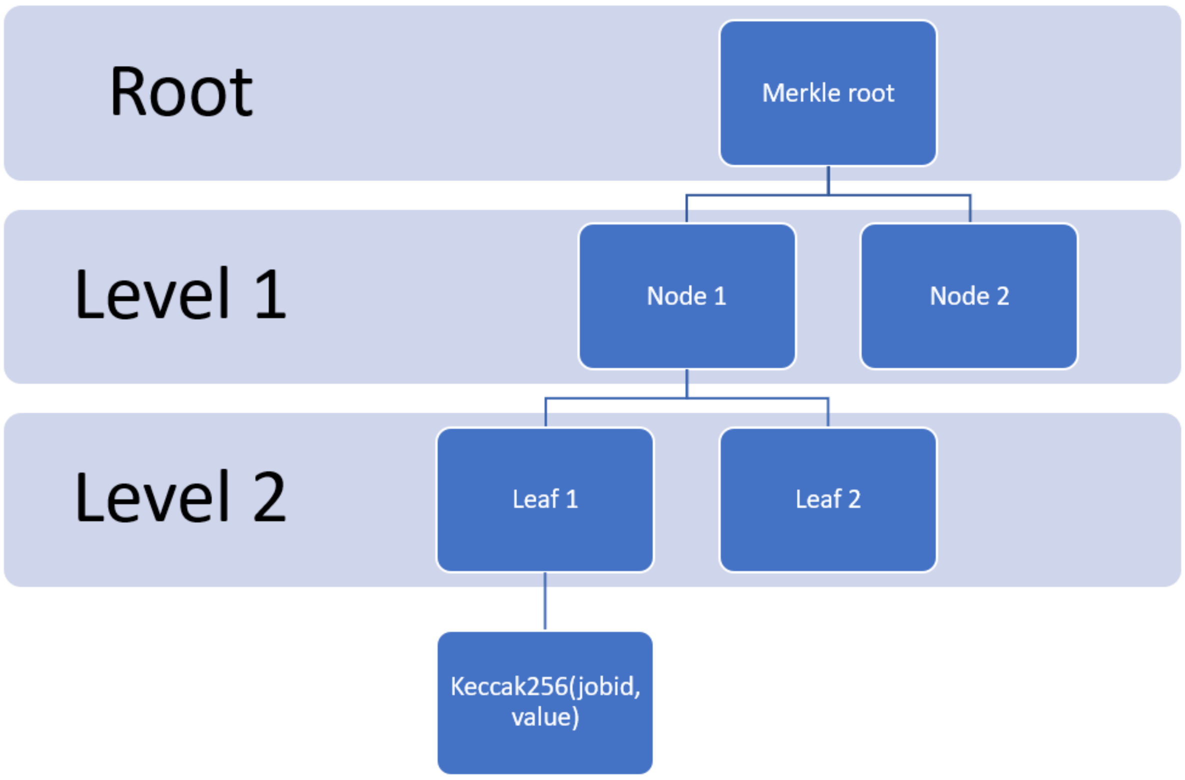

Figure 4: Merkle tree of commitments

[7] 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.

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)

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

$$H(e \mid\mid R \mid\mid S)$$

Here,

H = Collision Resistant Hash Function. We will be using keccak256.
e = epoch
R = Merkle Root
S = secret

Here, Secret is a 32 bytes random salt.

Stakers must 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.

2.4.4.3 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:

$$PRN = PRNG(B_C \mid\mid StakerId)$$

Where,

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

2. The following equation will be evaluated to evaluate which job is 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 nth from the top of the list
PRNG = 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.

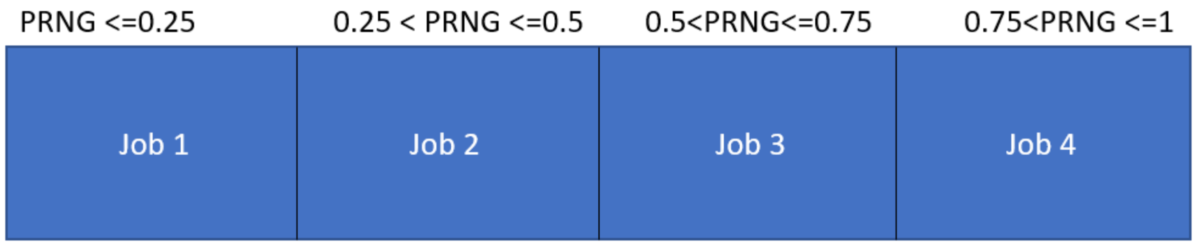

Figure 5: Assignment of jobs to a staker

If staker does not reveal during reveal state, she will be penalized.

2.4.4.4 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 list of stakers is pseudorandomly but deterministically calculable 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:

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

Where,

S_i = Staker ID
$PRNG$ = Pseudo Random Number Generator function which utilizes provided salt
n = nonce
B_R = blockhashes[8] 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 most influence

[8] 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.

The above steps are iterated and whenever it is true, that staker is added to end of the proposers list. Stakers who are already in the list are skipped.

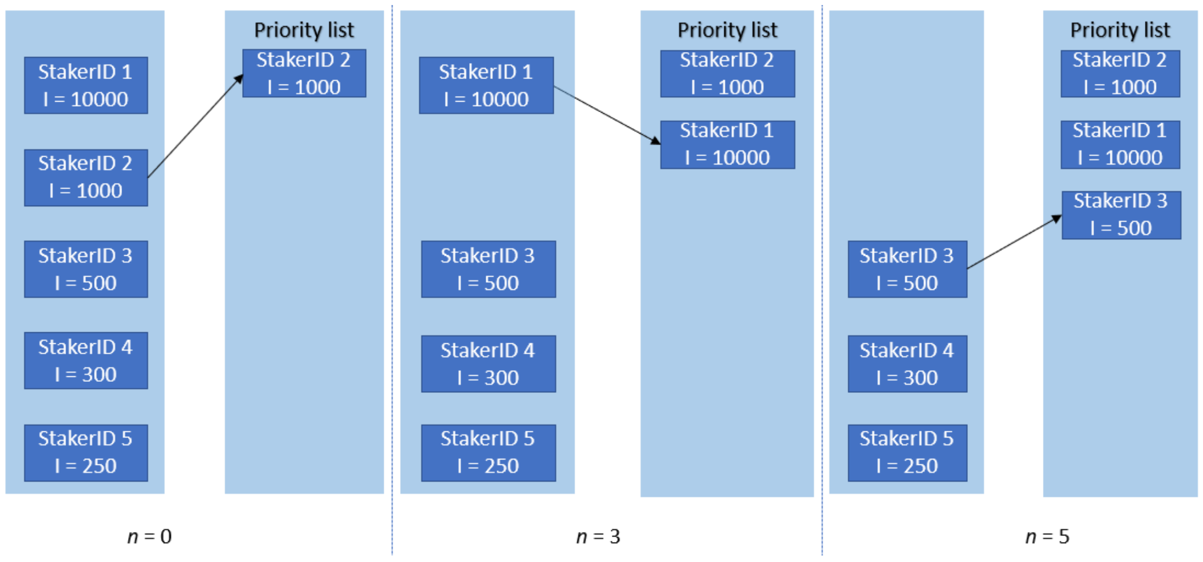

Figure 6: Selection for the block proposer list

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

2.4.4.5 Dispute Block

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

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

2.4.4.6 Unstake

If staker wants to withdraw stake, he 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).

2.4.4.7 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.

2.4.4.8 Submit results

This action submits a result once it is finalized to the requesting contract. This can be done during Commit state. The block proposer must perform this action or else she will not be awarded the block reward.

2.4.4.9 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.

2.5 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.

2.5.1 Influence

Razor network has points called influence points. Influence points determine a staker's influence on 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.

Whenever a staker performs stake action, she gets equal amount of influence points. Influence increases or decreases based on whether the staker's votes are in consensus with others. Influence points are earned by providing answers which are in consensus with answers provided by other stakers.

The following equation will be used to calculate Influence penalty percentage for stakers who report values outside consensus.

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

Where,

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

M = the weighted median calculated from all reported data points

For x > 2M, Influence penalty is 100%

Plot obtained from above equation is shown below. Let's assume M = 100.

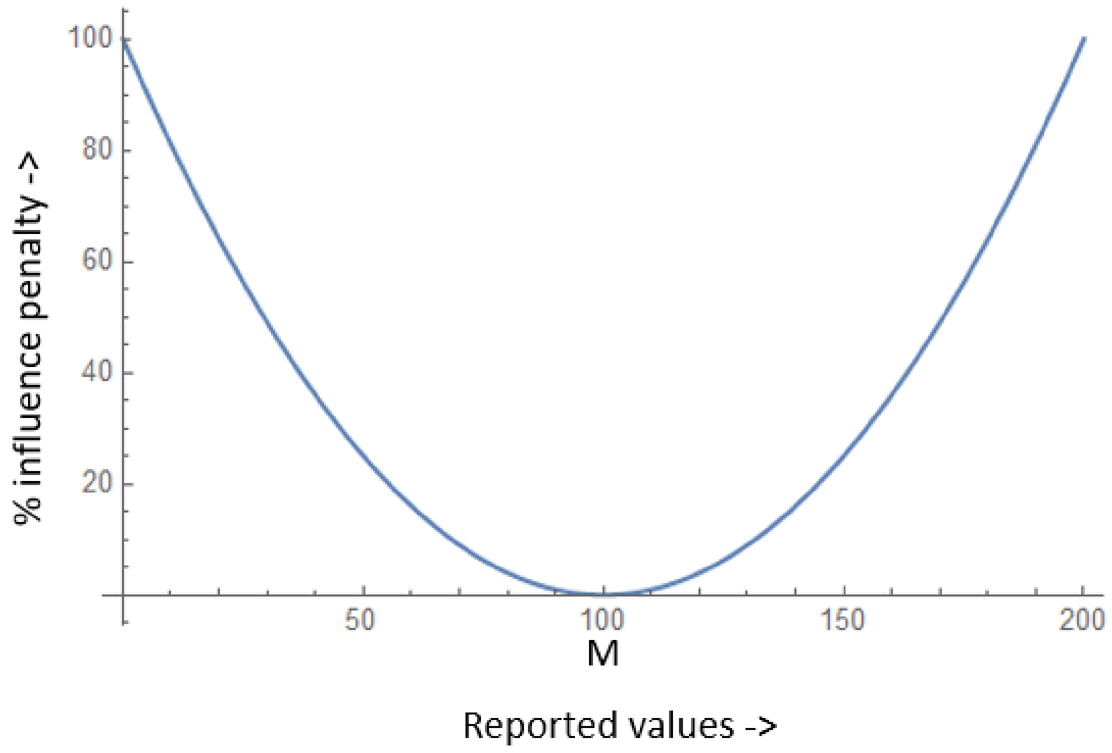

Figure 7: Influence penalty

2.5.2 Block reward

A block reward of B schells will be awarded to the stakers, if 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

2.5.3 Penalties for misbehaviour

Proposing invalid block: 100 % stake, 100% influence
Not committing in an epoch: 1% of influence
Committing but not revealing in an epoch : 5% of influence

2.6 Security

Widely used cryptographic primitives are used 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.

2.6.1 Economic security

Incentives are carefully designed to reward honest behaviour and punish malicious behaviour.

2.6.2 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.

2.6.2.1 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.

2.6.2.2 Takeover

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

2.6.2.2 Bribing

Bribing attack involves bribing the stakers, by the attacker, to vote in her favor. 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 high degree of economic security. Since it is impossible to know which stakers will be assigned to attacker's job in the commit state, attacker will need to bribe 51% of the network.

In addition to that, due to commit-reveal scheme used for reporting, trustless bribing attacks are impossible.

2.6.2.3 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.

2.6.2.4 Griefing

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

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 constantly changing. Oracles, being dependent on external data, must be flexible enough to adapt to the changing scenarios. External websites may become defunct, compromised, or behave in a byzantine manner. Since staking client 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 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 Criteria for whitelisting datasources

The stakers can decide to approve or reject data sources at their will. 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 not be legally, morally or technically “compromised”
4. Should preferably handle real asset versus derived asset (e.g. USD not USDT)
5. Reported values should not be outliers in any scenario
6. Should have sufficient liquidity, if it is an exchange
7. Should be confirmable from multiple independent data sources
8. Should respond reasonably fast
9. Reponses should not be too big (< 1 MB)
10. Should be freely accessible and should not require a login, proxy client, etc.

3.2 Voting

Voting can be done by active stakers. Voting can also be done by SCH holders who are not stakers by locking their tokens in the governance smart contract. The tokens stay locked in for a period after the voting has been concluded.

3.3 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)

3.4 Collections

Its a collection of data sources to provide a specific datapoint from various sources.
E.g. ETH/USD

3.5 Decisions on parameters of validation layer

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

3.6 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.

4 Applications

Any application which depends on real world data can utilize razor network to provide data points in a decentralized and trustless manner. Decentralized finance applications are especially suitable since they almost always require such a datasource.
We will explore some examples of such applications and explore how it can use Razor network.

4.1 Synthetic assets platform

1. This platform will use above two layers to create synthetic assets. Any collections on the governance platform can be used to mint assets.
2. Users can provide collateral to mint new assets according to price-feed values. Collateral can be sch, eth and sUSD (schelling usd). Schelling usd stablecoin will be a preferred collateral (except for ETHUSD) since it will likely be least volatile compared to all assets.
3. Users can burn assets anytime according to price-feed values to get back their collateral.
4. When assets are requested to be minted/burned the next future available price-point will be taken as reference.
    1. E.g. if I request to mint TSLA 10am, the last traded price at the beginning of next epoch will be used as reference.

5. When a position is under collateralized, anyone can liquidate a position by creating an update job for the oracle.
6. To long, buy a synthetic asset off the market. To short, mint it and sell it on market.