Zoracles: Confidential Oracles via Abstract Cryptographic Proofs

developers@zoracles.com

Abstract

When oracles were initially introduced into the crypto marketplace, the main challenges facing these products were connectivity and interoperability. As price feed data emerged as the clear use case for data provided by decentralized oracle networks, getting accurate pricing for crypto assets became a necessity to run many DeFi protocols that have billions of dollars in locked value.

Various oracle solutions have been deployed to facilitate connections between exchange data and smart contracts. Unfortunately they do not offer privacy guarantees to secure the flow of information from data sources. Therefore, we propose constructing zero-knowledge proofs and a random oracle model to provide confidential data to smart contracts. Our solution can be applied by enterprises to any smart contract platform by enabling the techniques described in this paper.

1. Introduction

Zoracles was designed to provide confidential data to smart contracts. We have developed our oracle solution by randomizing the sources of information and cryptographically constructing proving schemes and verifications for private data delivery.

Our approach can address the oracle problem of accessing off-chain data for smart contracts while providing a high degree of security using abstract proofs.

Most decentralized oracle networks haven't developed zero-knowledge proofs or any data privacy. Their focus has been creating adapters or cryptoeconomic incentives for staked validators.

Our view of the oracle mechanism in DeFi is fundamentally different than many projects. We view the software as a commodity that should perform an important service. Security, accuracy and reliability should be the foremost responsibilities of any oracle network. To date, none of the current solutions adequately address all of the aforementioned requirements.

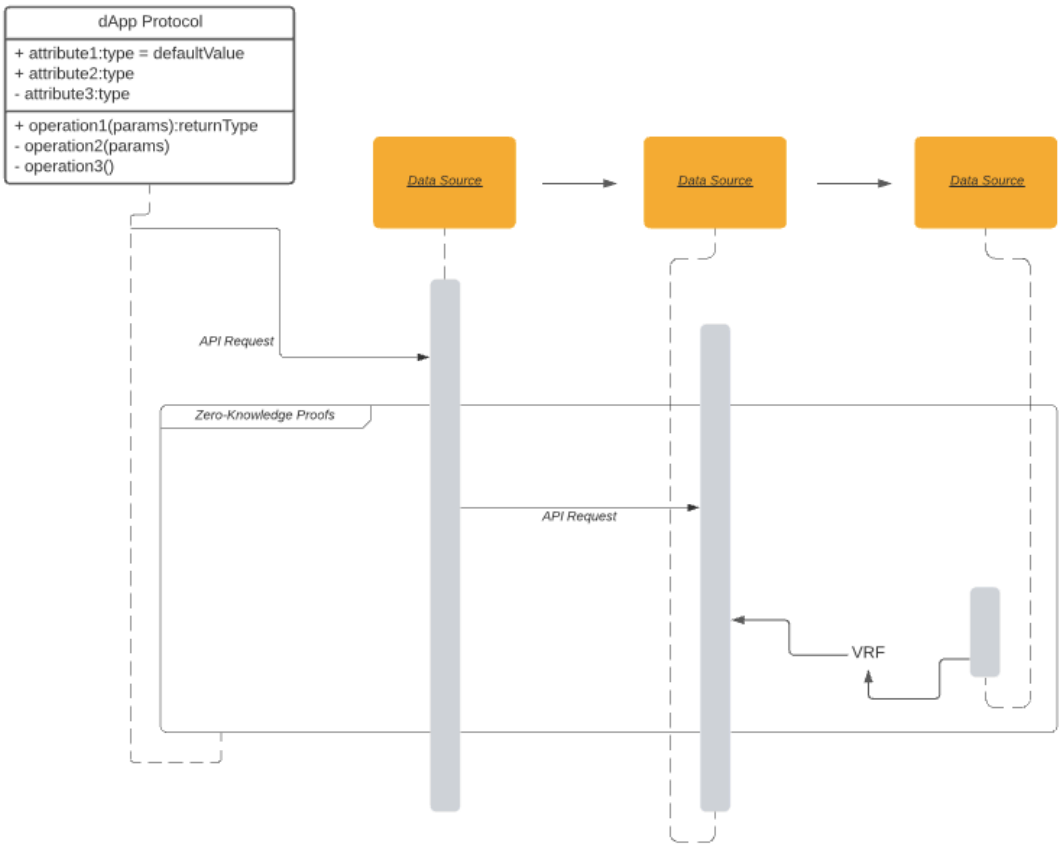
The focus on this paper will be detailing our end-to-end approach to securing oracle data and the cryptographic techniques required to provide strong privacy guarantees.

Specifically, Zoracles will leverage zero-knowledge proofs for confidential data delivery to smart contracts and verifiable random functions to randomize data sources. Applying these cryptographic capabilities to an open oracle standard will provide a comprehensive model that can be applied to various protocols.

Initially, when the oracle problem became an obstacle for smart contract functionality, the solutions developed focused on connectivity and interoperability for various blockchains. Our intention is to extend the functionality by providing privacy measures that can expand the use cases for smart contracts.

We believe the next step for smart contract adoption is supporting robust enterprise business processes. This can be achieved by applying cryptographic proofs to oracles that ensure their data is shielded from business competitors and outside manipulation.

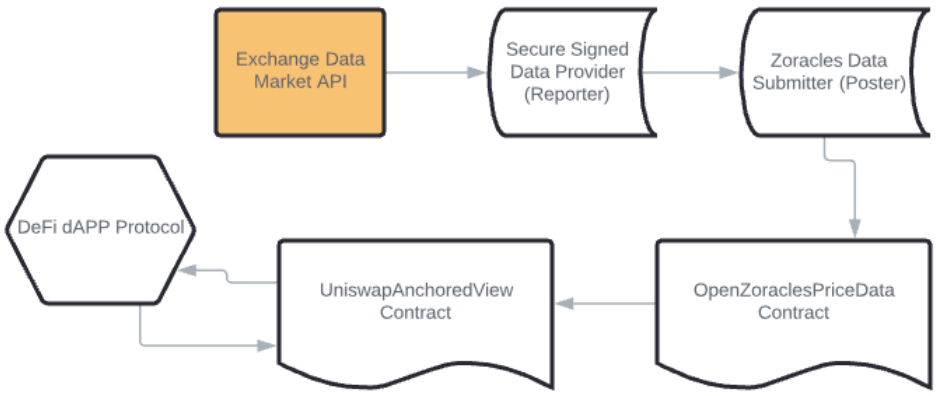
As shown below, our solution is an end-to-end design that leverages zero-knowledge proofs and a verifiable random function to achieve confidentiality of data.



2. Open Oracle

Open Oracle is a framework for retrieving off-chain data. We believe over time the industry will converge to a solution that has the least amount of friction and maximizes data security. Open Oracle offers both.

The structure of Open Oracle, as shown below, is divided between market data APIs, reporters/posters & dApp protocols. As demonstrated in the chart, the data flows from a given exchange (which is signed with a public key), to solidity contracts that will process the data on-chain with shielded inputs and finally used by a dApp protocol.

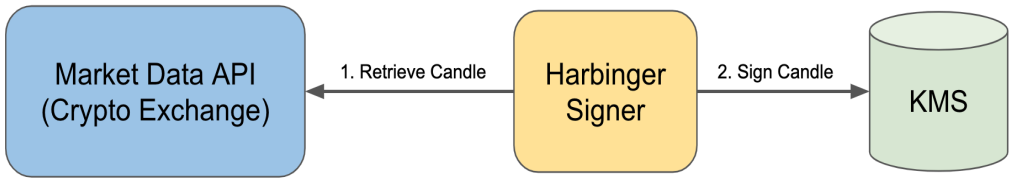


Compound released Open Oracle and Tezos [1] recently applied many attributes in their oracle solution. Our contribution will be developing strong privacy-preserving technologies that provide adequate security assurances for widespread enterprise adoption.

It is important to note that Open Oracle's mission is to provide a solution that many dApp protocols can adopt and standardize the process of retrieving off-chain data. Also, it can be used as a SDK and framework for building other data feeds such as weather, sports or any other API endpoint.

Open Oracle's design will be instrumental for future projects because they can easily tap into a trusted oracle solution that is currently used in production by large dApp protocols such as Compound and smart contract platforms such as Tezos.

The Tezos [1] Harbinger implementation of Open Oracle demonstrates the flexibility in the design with KMS. Zoracles is further extending the work of Open Oracle by building our solution within a similar framework. Moving forward, this SDK will allow for quick development and implementation by protocols needing secure price feed data.



It should not be understated that the simplicity in the design decisions by Open Oracle allows for any data feed to be considered for smart contract execution. DeFi projects can select different API endpoints, signers and reporters to plug into Zoracles while still maintaining the security benefits of our solution.

This will be attractive to enterprises that have billions of dollars to secure and need bank-level security measures to ensure confidentiality for clients.

Also, the Open Oracle framework provides for data signed by any public key to be considered in calculating a median value. The result can be aggregated and compared to a time-weighted average price that is determined by Uniswap v2. This measure protects the integrity of Zoracles Protocol against price manipulation.

2.1. Architecture & Implementation

Zoracles is furthering the development of Open Oracle with enhancements that aren't available with current oracle networks. We have applied zero-knowledge proofs to two main contracts to hide their input values:

- *OpenZoraclesPriceData** - this is where signed prices will be stored
- *UniswapAnchoredView** - prices are anchored to this time-weighted index to set boundaries on acceptable values and prevents adversarial influence

These additions will allow current Open Oracle users to upgrade these two contracts to leverage Zoracles additional privacy guarantees. Currently, there are projects in production that use Open Oracle that can easily redeploy them to improve security.

"*OpenZoraclesPriceData*"[2] is where price data is stored. Our solution will use these inputs to create a compact zero-knowledge proof that will be verified in "*UniswapAnchoredView*" contract where the output will be compared against a time-weighted average of Uniswap v2 prices.

The following function is where *getPrice* is called and will be used as inputs into our zero-knowledge proof. Once the verification proof is successfully executed, the dApp protocol can execute their smart contract.

```
function getPrice(address source, string calldata key) external view returns (uint64) {
    return data[source][key].value;
```

2.2. Reporters/Signers

A few key features of Open Oracle are the reporter and signer roles. A reporter can be a reputable exchange that will sign price data using their public key. Coinbase [3] is the first exchange providing this signed data and OKex will follow shortly.

Reporter Key	Supported Tokens
`0xfCEAdAFab14d46e20144F48824d0C09B1a03F2BC`	`BTC`, `ETH`, `DAI`, `REP`, `ZRX`, `BAT`, `KNC`, `LINK`, `COMP`

We expect more exchanges to eventually provide this signed data and this will significantly enhance the data security as described later in the paper.

Posters can be any dApp protocol with an Ethereum address that needs Zoracles confidential price feeds. We anticipate our strategic partnership with protocols to be the primary use case of the system and therefore become posters.

Our solution does not require a utility token or subject users to fees for API requests. Instead, posters pay gas fees associated with posting values on-chain and that will be significantly less than the cost structure of current oracle solutions.

As shown below, prices are posted to *OpenZoraclesPriceData & UniswapAnchoredView* in an array.

`messages[]`: The array of prices to sign. Each message is a `bytes[]` with the format: `(string memory kind, uint64 timestamp, string memory key, uint64 value)`

`signatures[]`: An array of signatures corresponding to the messages. Each signature is a `bytes[]` with the format `(bytes32 r, bytes32 s, uint8 v)`.

`symbols[]`: An array of symbols corresponding to the price messages.

2.3. Uniswap Anchor Pricing

An important feature of Open Oracle is the time-weighted average prices measured against Uniswap v2. This range is checked by "*UniswapAnchoredView*" contract to ensure it is within an acceptable range.

Prices are stored in "*UniswapAnchoredView*" with the expectation they have been previously verified by a zero-knowledge proof. The verification of price data with a compact proof is the core foundation of the Zoracles product.

Our design and development of a working constraint system with zero-knowledge proofs is the foundation of Zoracles value proposition and will be examined more thoroughly in this paper.

```
const view = UniswapAnchoredView.at(0xABCD...);
const res = await fetch("https://prices.compound.finance");
const {coinbase, okex} = await res.json();
const symbols = ['BTC', 'ETH', 'DAI', 'REP', 'ZRX', 'BAT', 'KNC', 'LINK', 'COMP'];
await view.methods.postPrices(coinbase.messages, coinbase.signatures, symbols).send({from:
```

A critical element of the Uniswap anchor pricing feature is the ability to normalize price movements of crypto assets across different exchanges into a common value. You can view the different assets and symbols in the figure above to gauge the range of supported symbols.

Employing a volume-weighted average of price value functions to protect data integrity against price manipulation. Zoracles Protocol needs built-in mechanisms that the data being transmitted is kept private, accurate, timely and free from outside influence of the normal market flow of data. This is accomplished with the Uniswap v2 price data balanced against the data issued and signed by reporters.

Open Oracle price feeds can support various token yield farming incentives provided that configuration of the metadata is set and immutable. The following chart [3] explains the fields required for dApp protocols. We expect this to attract significant usage to Zoracles protocol.

`cToken`: The address of the underlying token's corresponding cToken. This field is null for tokens that are not supported as cTokens.

`underlying`: Address of the token whose price is being reported.

`symbolHash`: The `keccak256` of the byte-encoded string of the token's symbol.

`baseUnit`: The number of decimals of precision that the underlying token has. Eg: USDC has 6 decimals.

`PriceSource`: An enum describing the whether or not to special case the prices for this token. `FIXED_ETH` is used to set the SAI price to a fixed amount of ETH, and `FIXED_USD` is used to peg stablecoin prices to $1. `REPORTER` is used for all other assets to indicate the reported prices and Uniswap anchoring should be used.

`fixedPrice`: The fixed dollar amount to use if `PriceSource` is `FIXED_USD` or the number of ETH in the case of `FIXED_ETH` (namely for SAI).

`uniswapMarket`: The token's market on Uniswap, used for price anchoring. Only filled if `PriceSource` is `REPORTER`.

`isUniswapReversed`: A boolean indicating the order of the market's reserves.

Zoracles framework will benefit from this feature by attracting innovative DeFi protocols that reward liquidity providers through issuing tokens. It should create a virtuous flywheel between dApp protocols, Zoracles and our “Snarks as a Service” structure and will be explored later in this paper.

3. Zero-Knowledge Proofs

A zero-knowledge proof or protocol is a way by which one party can prove to someone that they know a value without revealing any additional information aside from the fact that they are know the value [4].

Zoracles protocol provides confidential data to smart contracts with the construction of abstract cryptographic proofs that do not reveal any information to the receiving client or verifier.

3.1. Background

There are three properties, Zoracles protocol must satisfy to sufficiently develop zero-knowledge oracles [5]. The model below describes a general overview of the process to achieve anonymity without revealing any input data:

In this model the prover and the verifier are in possession of a reference string sampled from a distribution, *D*, by a trusted setup $\sigma \leftarrow \mathrm{Setup}(1^k)$. To prove statement $y \in L$ with witness *w*, the prover runs $\pi \leftarrow \mathrm{Prove}(\sigma, y, w)$ and sends the proof, π, to the verifier. The verifier accepts if $\mathrm{Verify}(\sigma, y, \pi) = \mathrm{accept}$, and rejects otherwise. To account for the fact that σ may influence the statements that are being proven, the witness relation can be generalized to $(y, w) \in R_\sigma$ parameterized by σ.

1) Completeness:

Generally, to satisfy completeness, Zoracles Protocol [5] verifying contract must accept the inputs provided by a data source:

Verification succeeds for all $\sigma \in \mathrm{Setup}(1^k)$ and every $(y, w) \in R_\sigma$.

More formally, for all *k*, all $\sigma \in \mathrm{Setup}(1^k)$, and all $(y, w) \in R_\sigma$:

$$\Pr\left[\pi \leftarrow \mathrm{Prove}(\sigma, y, w) : \mathrm{Verify}(\sigma, y, \pi) = \mathrm{accept}\right] = 1$$

2) Soundness:

Generally, to satisfy soundness, Zoracles Protocol [5] rejects any incorrect inputs outside an insignificantly small probability:

$$\Pr\left[\sigma \leftarrow \text{Setup}(1^k), (y,\pi) \leftarrow \tilde{P}(\sigma) : y \notin L \wedge \text{Verify}(\sigma, y, \pi) = \text{accept}\right] = \nu(k)\ .$$

3) Zero-Knowledge:

Generally, to satisfy a zero-knowledge proving system, Zoracles Protocol [5] must not learn any information about the data sources except that the data provided is accurate and correct:

$$\Pr\left[\sigma \leftarrow \text{Setup}(1^k) : \mathcal{A}^{\text{Prove}(\sigma,\ldots)}(\sigma) = 1\right] \equiv \Pr\left[(\sigma,\tau) \leftarrow \text{Sim}_1 : \mathcal{A}^{\text{Sim}(\sigma,\tau,\ldots)}(\sigma) = 1\right]$$

Here $\text{Sim}(\sigma,\tau,y,w)$ outputs $\text{Sim}_2(\sigma,\tau,y)$ for $(y,w) \in R_\sigma$ and both oracles output *failure* otherwise.

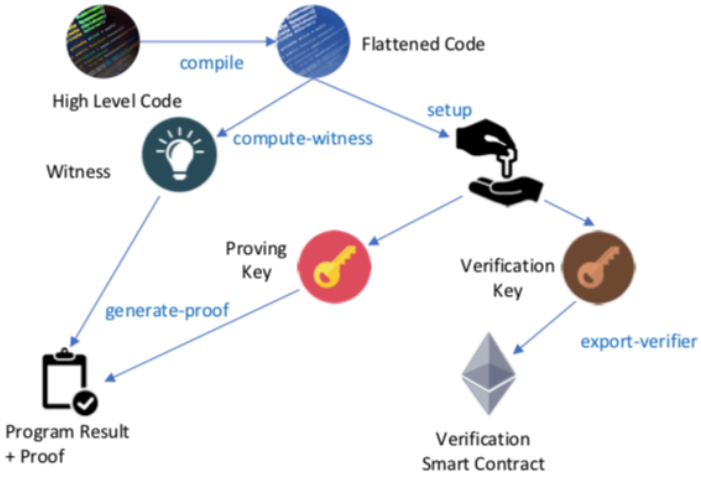
There is an additional challenge working with zero-knowledge oracles. Values stored in the view contract need to be homorphically hidden to retain privacy for the entire data delivery process.

This presents an interesting obstacle that was addressed as a main ingredient of zero-knowledge proofs in Reitweibner's paper [6] in encoding values as a polynomial problem. "The program that is to be checked is compiled into a quadratic equation of polynomials: t(x)h(x) = w(x)v(x), where the equality holds if and only if the program is computed correctly. The prover wants to convince the verifier that this equality holds."

Zoracles protocol will homorphically hide exchange values with an encryption function that the verifier will use to satisfy all the elements of a zero-knowledge proof. This step is critical in forming an end-to-end solution that is privacy preserving.

3.2. Zokrates Toolbox

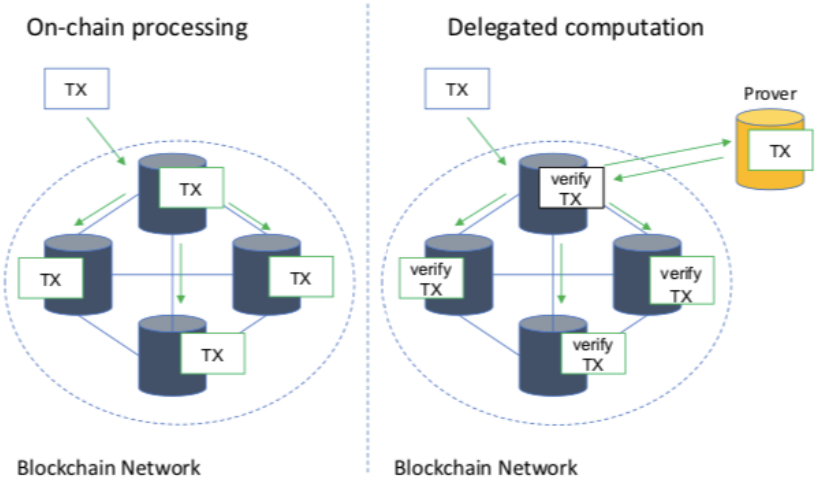
An important component of Zoracles speed of development in deploying zero-knowledge proofs on smart contracts will be building with the abstract language provided by the toolbox within ZoKrates [7].



The use of abstract cryptography to create probabilistic proofs with ZoKrates is a powerful way to compile flattened code and generate proving keys for API data and verification contracts for smart contracts.

Moreover, leveraging ZoKrates ensures that our proofs satisfy key attributes of Zk-Snarks including succinctness, non-interactivity and most importantly the platform provides a quick way to deploy arguments of knowledge.

Using ZoKrates toolbox allows Zoracles development team the freedom to concentrate on designing the programs that are useful functions in building Snarks. The process paves the way for clients to take advantage of off-chain computations and leverage the privacy features of keeping sensitive data shielded from the blockchain.



Zoracles developers have created a proof of concept with ZoKrates to determine the price of an asset like gold, silver and Bitcoin in a particular range.

Zero-knowledge proofs can be used to provide stronger privacy guarantees. Without disclosing the price and the source of the data, these crypto assets spot pricing can be verified by a client.

Motivation

These price range proofs can be used in some of the following financial applications: prediction markets, binary options trading, auctions, etc. These applications can leverage the price range proof to verify that the price of an asset is between a certain range of values defined by MinAndMax.

Proof Definition

We will generate a Zk-SNARK proof to verify these constraints. The advantage with this approach is that the proof ensures privacy and the size of the proof is small compared to some other zk proofs. The private input in our case is the price of the asset. This price is ideally provided by some trusted sources.

The MinAndMax price will be the public inputs to the Snark. The constraints:

1. `price` > `min`
2. `price` < `max`

These constraints are enough to prove that the price of an asset is in the provided range.

Prover

The proof is generated by the prover after a trusted setup, so that fake proofs can't be generated. This proof can be used to verify arbitrary values of price, min, max. Before generating the proof, the prover needs to generate the witness from the inputs. New witness and proof have to be generated each time for new input values.

Verifier

The verifier in this proof of concept will be a smart contract which is deployed on Ethereum. This verifier will only be generated once. This verified can be used to verify any proof given the public inputs (MinAndMax).

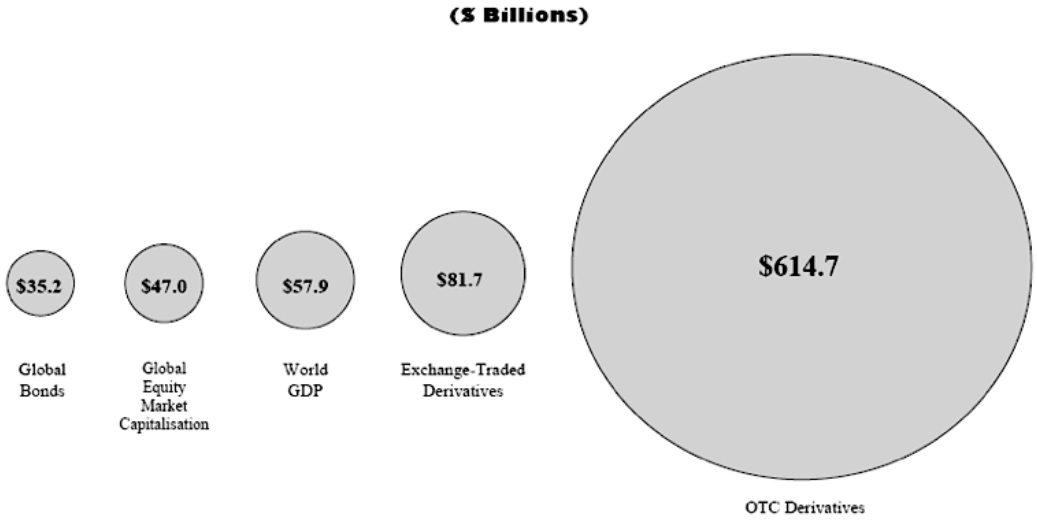
Code

To generate the zero-knowledge proofs, we use Zokrates abstract language for specifying the function to perform the simulation:

```
def main(private field price, field min, field max) -> bool:
  assert(price>min)
  assert(price<max)
  return true
```

This proof of concept is a snippet of the powerful programmability of ZoKrates to deploy Zoracles for the global derivates market.

The global derivatives market is growing and will definitely require confidentiality for trading billions of options by the minute. We can expand and create more complicated Snarks based on the specific business requirements reflecting the core value proposition of the protocol.



3.3. Snarks As A Service

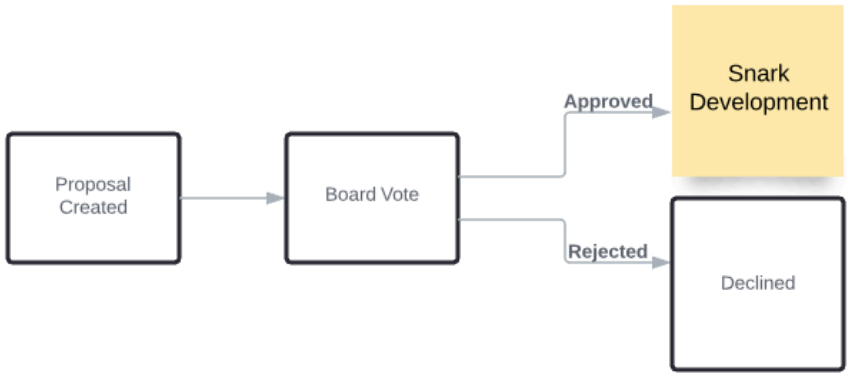
The aforementioned proof of concept is an important representation of the opportunity with building custom Snarks for enterprise clients. They will require the highest level of confidentiality with trillions of dollars at stake.

Our governance is designed to form a tight partnership between individuals or enterprises building on blockchains that need strong privacy guarantees. They can acquire a stake in our governance to influence our development team on building a Snark that fits their business objectives.

Zoracles governance is designed and modeled after a corporate board. You need to acquire at least 1 token ($Zora) to make a proposal that will be reviewed by our board.

The goverance of the protocol is inspired by Bitcoin's "1 cpu = 1 vote [8]"

Zoracles is "1 Zora = 1 Vote."



4. Random Oracle Model

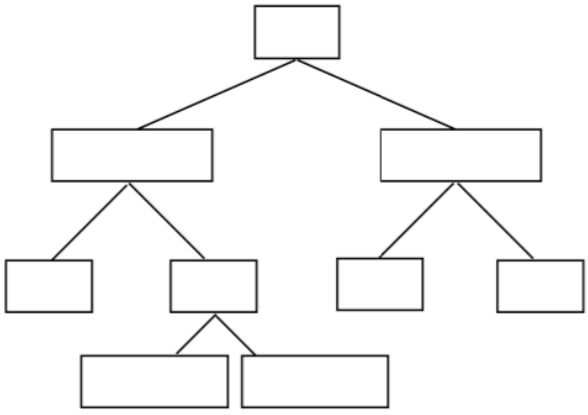
A large attack vector that persists for smart contract execution is oracle data. Price manipulation is especially undesirable due to the unwanted influence potentially undermining the integrity of automated financial agreements. When large amounts of locked value is at stake, it is imperative that the high level of security measures are employed to secure the flow of information.

4.1. VRF

A verifiable random function (VRF) according to Micali, Rabin & Vadhan [9] needs to fulfill the following criteria:

1. a compact, implicit representation, which does not enable one to evaluate a function efficiently, and
2. a compact, explicit representation, which enables anyone to evaluate a function efficiently

This can be visually represented [9] in a tree construction:

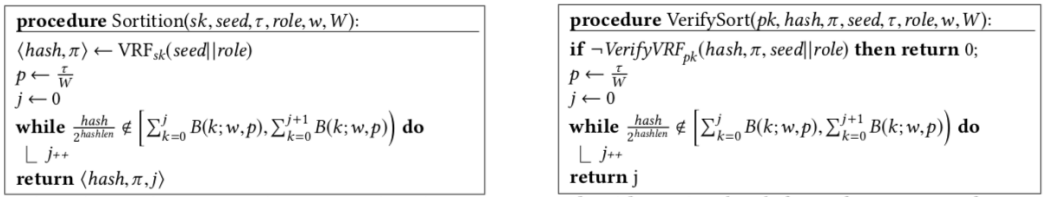


Zoracles is in the very early exploration phase of building VRFs to further strengthen privacy by obscuring data sources. This work should be classified as experimental but we believe our end-to-end solution will incorporate VRFs.

4.2. Data Sources

The foundation of any reputable oracle network is its data. The ability to provide consistent and accurate information to a client represents the core value proposition by the oracle mechanism.

Zoracles proposes using a VRF similar to the published Algorand [10] sortation algorithm to randomize the API endpoints to significantly enhance the security against price manipulation by utilizing the following function:



Zoracles proposes using VRFs [10] to randomize the API endpoints to significantly enhance the security against price manipulation.

Based on the work of Hoeteck Wee, [11] we expect rounds of interaction between the prover and verifier to achieve zero-knowledge in the random oracle model. Further research is necessary to validate this assumption.

5. Conclusion

Using powerful probabilistic proofs, the Zoracles protocol is able to provide strong privacy guarantees for DeFi data input into smart contracts.

Our initial focus has been building proofs within the abstract language of ZoKrates. Long-term the development team will need to expand and design custom circuits to provide more robust solutions and "Snarks as a Service."

We are also exploring implementing VRFs as a mechanism to randomize data sources. This would be a significant improvement in masking the API endpoints for facilitating confidential data delivery for dApp protocols.

Overall, we have devised a clear roadmap for an end-to-end solution for enabling privacy on Open Oracle. Our Zk-Snarks programs can be applied immediately to the derivatives market and future work will consider NFTs as an expansion opportunity.

Further research will significantly enhance our protocol with VRFs implementations and allow Zoracles Protocol core product to get adapted by developers and enterprises.

References

[1] Martin Young, "Tezos improves DeFi infrastructure with Harbinger price oracle," https://cointelegraph.com/news/tezos-improves-defi-infrastructure-with-harbinger-price-oracles

[2] Zoracles Github, https://github.com/zoracles/zora/blob/master/OpenZoraclesPriceData

[3] Compound Open Price Feed, https://compound.finance/docs/prices#introduction

[4] Wikipedia. https://en.wikipedia.org/wiki/Zero-knowledge_proof"

[5] Wikipedia. https://en.wikipedia.org/wiki/Non-interactive_zero-knowledge_proof

[6] Christian Rietwiebner,"zkSnarks In a Nutshell," http://chriseth.github.io/notes/articles/zksnarks/zksnarks.pdf

[7] Jacob Eberhardt, Stefan Tai, "ZoKrates –Scalable Privacy-Preserving Off-Chain Computations," https://www.ise.tu-berlin.de/fileadmin/fg308/publications/2018/2018_eberhardt_ZoKrates.pdf .

[8] Nakamoto, Satoshi, "Bitcoin: A Peer-to-Peer Electronic Cash System," https://bitcoin.org/bitcoin.pdf, October 31, 2008

[9] Micali, Silvio, Rabin, Michael, Vadhan, Salil, "Verifiable Random Functions," https://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Pseudo%20Randomness/Verifiable_Random_Functions.pdf

[10] Gilad, Yossi, Hemo, Rotem, Micali, Silvio, Vlachos, Georgios, Zeldovich, Nickolai, "Algorand: Scaling Byzantine Agreements for Cryptocurrencies," https://dl.acm.org/doi/pdf/10.1145/3132747.3132757

[11] Hoeteck Wee, "Zero-knowledge in the Random Oracle Model, Revisted, https://www.iacr.org/archive/asiacrypt2009/59120414/59120414.pdf