Research Proposal
An Idea of an Proof Based Decentralised Distributed Oracle Protocol

Wenxing Duan

November 11, 2022

Abstract

This paper aims to propose an idea for a new oracle protocol design. First, this article introduces the historical process and basic principles of the blockchain system. Then it discusses the importance of oracles and introduce some current attack methods against existing oracle systems. This article divides the main oracle systems in the field into five categories, expounds the principles and potential risks, providing a comparison. Afterwards, I envisioned potential improvements to the design of one of the oracles. At the same time, combined with the latest research progress of zero-knowledge proof, I also provide a prototype of a possible solution.

1 Introduction

In 2008, Satoshi Nakamoto released bitcoin whitepaper: "Bitcoin: A Peer-to-Peer Electronic Cash System" [1]. With the creation of the Bitcoin Genesis block, the blockchain was born. In the following thirteen years, the blockchain technology has flourished. In the design of Bitcoin, the role of the blockchain is simply to implement a cryptographically protected wallet account system based on cryptography and proof of work algorithm. In 2014, a blockchain system with functionality of executing the code stored the blockchain was created, which is Ethereum [2]. Since the creation of Ethereum, the blockchain technology is no longer just for token transaction, but has evolved into a world computer. This is why people often refer to the blockchain system represented by Bitcoin as Blockchain 1.0 and the system represented by Ethereum with smart contracts as Blockchain 2.0.

Due to the security and privacy of blockchain wallets, decentralised finance (De-Fi) is growing rapidly. They are deployed on blockchain systems like Ethereum in the form of smart contract code and provide financial services such as loaning, exchange, etc. to every blockchain user. According to statistics, a total of approximately $27 billion of cryptocurrency has been locked in De-Fi to date. How to guarantee the security of these funds becomes one of the main concerns of blockchain researchers.

2 Background

Because of the nature of the blockchain system, it makes it very difficult for the smart contract code on the chain to access data on the external internet. Almost all smart contracts must rely on an Oracle machine to access external data. For De-Fi systems in particular, the accuracy of the price data provided by the prophecy machine has a direct impact on the safety of the user's money [3]. And not only for De-Fi system, Oracle data integrity is also important for Decentralized Identity authentication like CanDiD [4], which is significant for DAO system. How to provide Oracle data provisioning services quickly, cheaply and accurately has become a primary goal for many blockchain scientists.

Currently, most of the oracle system designs fall into five main categories.

- **Optimistic Oracle Model**
  The first and simplest is the optimistic Oracle machine represented by the UMA [5]. This kind of Oracle machine chooses to trust provider's data optimistically and encourages anyone else to dispute the accuracy of this data. Anyone who successfully disputed will get rewards and data provider will be punished. The cost of this kind of Oracle machine is very low, but it could still face a Sybil attack or bribery attack. Also, the timeliness is low, cause for every data requires a time slot for others to check and raise disputation.
- **Majority Consensus Oracle Model**
  The second type of Oracle machine uses a reputation system and the majority principle, represented by Chainlink1.0 [6] and Witnet [7]. This Oracle machine chooses to believe to be the one offered by the majority of all data providers with

a high reputation or selected by reputation, while reducing the reputation of data providers who Oracle believes have provided incorrect data. The security of this kind of design is significantly better than the first one mentioned above, with the resistance of bribery attack and DoS attack, also with high timeliness and low gas cost. But it could still face the danger of Sybil attack and freeloading attack which is some data providers simply copying other data providers' data to get the rewards.

- **Game Theory Based Oracle Model**
  Third kind is represented by NEST [8], based on game theory, using arbitrageurs to ensure the accuracy of token price. By asking the token price data provider to lock amount of the token and USDT into the system, allow other arbitrageurs to trade tokens or USDT with the price they provided, and ask the trader to provide a new price and deposit more tokens and USDT than the previous provider, it creates a price chain in the system, whenever there is an inaccuracy in the price, arbitrageurs flock to correct the price to the accurate value. The Adversary needs a significant amount of money to beat everyone in the network to control the price offered by the Oracle machine. This Oracle machine provides the most accurate price data with the highest stability. But it is limited in that it can only provide price data and cannot provide any other type of data.

- **Trusted Environment Oracle Model**
  Forth type of Oracle machine is based on the trusted computing environment, represented by Town Crier [9] and Android proof in Provable [10]. By providing proof of the correctness of the running procedure of code in the safe environment like Intel SGX, users could verify that the data provider is collecting data from the correct data source. The security mechanism highly relies on the security of the safe environment, but more and more researches show that attack on safe environment itself is achievable [11].

- **Cryptographic Proof Based Oracle Model**
  Last kind of Oracle machine is rely on the cryptographic proofs over TLS [12] protocol, represented by DECO [13] and TLSnotary [14] in Provable. Through cryptographic methods, the verifier and the data provider jointly hold a portion of the TLS session key, ensuring that the message is tamper-proof and allowing the data provider to prove the integrity of the data. But the process of three-party handshaking and verifying proofs is expensive, which is why in Chainlink 2.0 [15], although DECO has been deployed, it does not verify every data feeding, but only in Second-Tier adjudication event. Also, this design has a huge drawback in that only the verifier at the beginning of the protocol can verify this proof, and no other third party can. This means that verifiers and data providers can jointly forge fake data to provide to third parties.

3 Hypothesis and Objectives

I summarize the characteristics that should be satisfied for an ideal oracle and compare the systems mentioned above in the figure 1.

	Optimistic Oracle Mode	Majority Consensus Oracle Model	Game Theory Based Oracle Model	Trusted Environment Oracle Model	Cryptographic Proof Based Oracle Model
Not reliant on data source accuracy	■	■	■	■	■
Multiple data type support	■	■	■	■	■
High timeliness	■	■	■	■	■
Sybil Attack resistance	■	■	■	■	■
DoS Attack resistance	■	■	■	■	■
No need for specific hardware	■	■	■	■	■
No need for specific communication protocol	■	■	■	■	■
Low gas cost	■	■	■	■	■
Integrity can be verified by anyone	■	■	■	■	■
High attack difficulty	■	■	■	■	■

Figure 1: Summary and Compare

It can be obvious that only some types of oracle systems can satisfy the most ideal oracle model due to the limitation of the on-chain program. No oracle protocol can satisfy all desirable properties, and for the vast majority of current blockchain projects, such a perfect oracle is not necessary. However, for the sake of data integrity and accuracy and the stability of oracle operation, a proof-based oracle system with resistance to various attacks is urgently needed for the current software ecosystem. In my hypothesis, it is possible to construct an oracle with the following characteristics:

- **Universal proof of data integrity**
  For the current Cryptographic Proof Based Oracle Model, the biggest shortcoming is that only the verifier at the beginning can verify the proof provided by the prover because only the verifier holds the other half of the TLS Session key. Because the user in this context refers to the on-chain code, the user does not have the ability to act as a validator since storing the other part of the session key on-chain will be visible to everyone. Therefore the verifier is usually deployed on the server of the oracle provider. For Provable oracles, for example, the verifier code is deployed on AWS. At this time, Provable acts as both a verifier and a prover. If an adversary hacks Provable's AWS servers, the adversary can forge TLS communication packets and generate the proof of the fake data for the user. Or if Provable finds that the information requested by the user has a huge arbitrage opportunity, Provable can also send the wrong information without being detected. Although Provable runs auditors in AWS and regularly publishes code auditor information to convince users, it is possible to forge fake auditor information.
  The ideal blockchain system is a zero-trust system. Users should be able to do what they want without trusting anyone. Therefore, I envision an oracle system that can provide universal proof. Anyone can participate in the role of a verifier and verify the proof given by the prover in a TLS runtime.

- **No need for trusted environment**
  An oracle system that relies on a trusted environment is not a good idea. On the one hand, it increases the operating cost of the oracle system operator. On the other hand, once a vulnerability is discovered in a trusted environment, the proof systems of all oracles using this trusted environment will no longer be trusted.

- **Sybil Attack resistance**
  Sybil Attack on Oracle is adversary forge or controls a large number of fake nodes to control the outcome of the oracle. This kind of attack is specifically aimed at the Majority Consensus oracle model, and by increasing the number of a node could significantly increase the cost of the rising Sybil attack. But when the potential arbitrage benefits of forging this information are greater than the cost of controlling the majority of oracle nodes, an attack still could happen
  In the oracle, I imagined, based on a complete proof system, the difficulty of the adversary forging data lies not in controlling the majority of nodes but in generating data integrity proofs for fake data, which will exponentially increase the adversary's attack cost.

- **Timeliness and DoS Attack resistance**
  Under the Cryptographic Proof Based Oracle Model or Trusted Environment Oracle Model, the timeliness of the data is very high. But they may be subject to denial of service attacks by node operators or nodes servers that have been hacked. Denial of Service attack on oracle system doesn't necessarily means stopping the whole system. Instead, simply cutting the service of a specific data requested for a few minutes is enough to cause a considerable impact. In Cryptographic Proof Based Oracle Model and Trusted Environment Oracle Model, although the prover and verifier don't known the content of the TLS query, the destination of the packet is clear. If some of the nodes are run by a malicious party, they could raise a DoS attack and cause a delay on a certain type of data (for example, sports game result for gambit protocol), which could cause serious harm.
  This attack would theoretically work for proof-based oracles, but it would be difficult for the distributed oracle system like Chainlink. My vision is to build a distributed proof-based protocol. In this scenario, all oracle nodes can provide proof for the data. Instead of running DECO or Town Crier on nodes separately, all nodes collectively provide proof for a single query. In the ideal imagined oracle, as long as one node is behaves honestly, a DoS attack cannot work.

4 Methodology

I proposed two ideas for oracle models that can achieve the above goals, one is based on the modification and improvement of the DECO protocol, and the other is based on a STARK-based programming language called Cairo [16].

4.1 Improved Cryptographic Proof Based Oracle Model

Although DECO can set the verifier and prover to be distributed which everyone can take part in, to prove that it cannot be both a verifier and a prover to provide fake data proof. The adversary can still control the oracle by setting up a large number

of verifier and prover nodes, which also makes DECO lose its ability to resist Sybil attacks. In order to avoid this, I imagined that the verifier side combines the secret sharing, so that multiple different verifiers can hold the session key for the versifier end at the same time, and only the cooperation of a specified number of verifiers can reveal the real session key.

- **Secret Sharing - Blakley's scheme [17]**
  The principle of Blakley's scheme is simple. For example, a secret needs to be shared with k entities, and only n of the k entities need to be combined together and reveal the secret ($0 < k \leq n$). Firstly, encode the secret as a coordinate in n-dimensional space, then generate k number of nonparallel hyperplanes with a dimension of $n-1$ that passed the secret coordinate, and distribute to k entities. Under this scenario, to revel the secret (coordinate), any n entities can combine their hyperplanes and be found out the intersection of the hyperplanes.

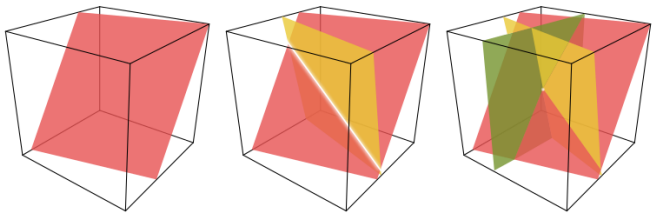

Figure 2: (Image from Wikipedia [18]) An example of $n = 3$, it can be found that when only two planes are involved, the secret coordinates cannot be revealed.

In DECO, the verifier generates its own public key and sends it to the prover. The prover combines the verifier's public key with the public key generated by itself to form a public key for communication then sends it to the server. When the session key encrypted by the server with the combined public key is received, the session keys of the verifier and the prover are distributed separately through the $\mathcal{F}_{2PC}^{hs}$ protocol. By adding the secret sharing protocol to the $\mathcal{F}_{2PC}^{hs}$ protocol, the secret sharing of the session key can be easily realized.

Assuming that the user specifies k verifiers for secret sharing, at least n verifiers are required to reveal the key. Suppose that among all verifiers, the malicious verifier ratio is m. It can be found that at least n/k of the verifier is malicious, and the fake data integrity proof can be generated. As long as for the choosen n and k, $n/k > m$, the proof system is secure.

However, an obvious disadvantage is that this greatly increases the chances of a DoS attack. For both the verifier and the prover, the URL of the website the user is trying to get the data from is visible. Suppose there is an adversary among the n verifiers. In that case, he can interrupt the communication by not participating in the secret disclosure. Then the system needs to re-select or wait for the new n verifiers to participate in the secret revealing process. And the probability of a situation where there is no adversary in n is $(1-k)^n$, which could be very small if n is chosen to be large for safety. I envision the Tor network system being used here to address this potential risk.

- **Tor [19]**
  Tor is an anonymous communication service based on onion routing. Tor is divided into three parts: entry node, relay node and exit node. Tor uses the layer-by-layer encryption of these three-layer nodes, so that no other node knows the data information and destination of this communication session except the exit node.

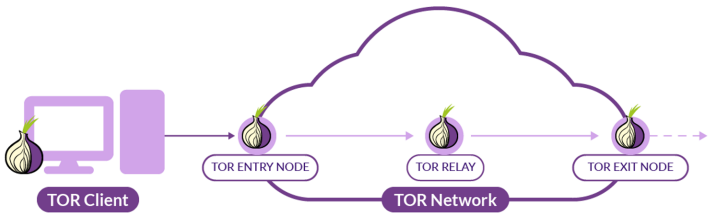

Figure 3: (Image from sysdig.com) A schematic diagram of Tor

I envision that instead of directly asking the oracle for the data, the user requests a Tor communication from the oracle. Before sending the call to the oracle, user generate the query packet for the relay layer in Tor and sends the packet with the entry node address to the oracle. Oracle communicates with the entry node, waits for respond, and then delivers to the user.

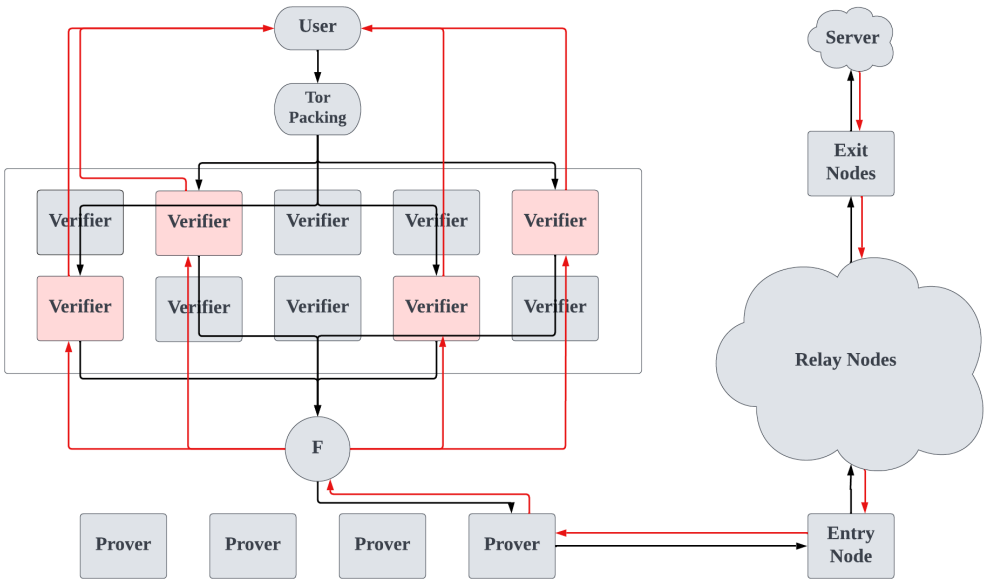

Figure 4: Schematic diagram of the model proposal, black line represents sending, red line represents returning

4.2 Provable Programming Language Approach

The idea is still in its infancy. Cairo is a programming language for writing provable programs currently used in the practice of ZK-Rollup. By executing transactions and codes at Layer 2 and providing corresponding state proofs for program verification and synchronization on the chain, ZK-Rollup technology theoretically greatly improves the processing speed of Ethereum. Researchers have successfully compiled UniSwap's smart contract code into Layer 2 StarkNet, named UniStark [20]. This technique of proving the correctness of a program may be used in scenarios such as oracles that require proof of data integrity. Proving that the data did come from the specified link from the perspective of the virtual machine and programming language, rather than through the encryption process of TLS, which may be a solution. But I still need more study to explore the feasibility of this scheme.

5 Summaries and Limitations

In summary, I introduced the current mainstream solutions for oracle systems and analyzed their advantages and disadvantages. In response to these shortcomings, I conceived the characteristics of an ideal high-security oracle model, and proposed possible improvements for one of the oracle systems. At the same time, according to the latest progress in the field of zero-knowledge proofs, I also imagined another completely new solution, but this is still a naive conception and needs more studies and researches.

The improvements I propose for DECO still have many shortcomings. First, such a design would significantly increase operating costs. At the same time, because the speed of the Tor network is very slow, the method of using Tor for anonymous communication will also increase the data delay. While the current design of the verifier is resistant to profit-seeking DoS attacks, the idea of allowing everyone to act as a verifier is hard to stop an adversary seeking sabotage from participating in the verification and launching the attack. More study and research are still needed for these potential shortcomings and risks.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[2] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37):2–1, 2014.

[3] Torgin Mackinga, Tejaswi Nadahalli, and Roger Wattenhofer. Twap oracle attacks: Easier done than said? *Cryptology ePrint Archive*, 2022.

[4] Deepak Maram, Harjasleen Malvai, Fan Zhang, Nerla Jean-Louis, Alexander Frolov, Tyler Kell, Tyrone Lobban, Christine Moy, Ari Juels, and Andrew Miller. Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1348–1366. IEEE, 2021.

[5] Universal Market Access. *UMA Protocol: How does UMA's Oracle work?*, 2022.

[6] Steve Ellis, Ari Juels, and Sergey Nazarov. Chainlink: A decentralized oracle network. *Retrieved March*, 11:2018, 2017.

[7] Adán Sánchez de Pedro, Daniele Levi, and Luis Iván Cuende. Witnet: A decentralized oracle network protocol. *arXiv preprint arXiv:1711.09756*, 2017.

[8] nestprotocol.org. Nest decentralized probabilistic virtual machine model. 2022.

[9] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. Town crier: An authenticated data feed for smart contracts. In *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*, pages 270–282, 2016.

[10] provable.xyz. Android proof: Authenticated data gathering using android hardware attestation and safetynet. 2019.

[11] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient {Out-of-Order} execution. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 991–1008, 2018.

[12] Tim Dierks and Eric Rescorla. The transport layer security (tls) protocol version 1.2. Technical report, 2008.

[13] Fan Zhang, Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. Deco: Liberating web data using decentralized oracles for tls. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1919–1938, 2020.

[14] Pawel Szalachowski. Blockchain-based tls notary service. *arXiv preprint arXiv:1804.00875*, 2018.

[15] Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs*, 2021.

[16] Lior Goldberg, Shahar Papini, and Michael Riabzev. Cairo–a turing-complete stark-friendly cpu architecture. *Cryptology ePrint Archive*, 2021.

[17] George Robert Blakley. Safeguarding cryptographic keys. In *Managing Requirements Knowledge, International Workshop on*, pages 313–313. IEEE Computer Society, 1979.

[18] Wikipedia contributors. Secret sharing — Wikipedia, the free encyclopedia, 2022. [Online; accessed 10-November-2022].

[19] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[20] Jorik Schellekens. Introducing unistark: Uniswap, only warp'ed to starknet. 2022.