# WINkLink

## A decentralised oracle network on TRON

1 July 2023 (v2.0)

**Abstract**: Smart contracts are one of the most important parts of modern blockchains. Smart contracts are deployed on blockchains, triggered automatically, and cannot be modified after deployment. These characteristics make smart contracts the best solution for traditional digital contracts. However, smart contracts cannot communicate with data outside the blockchains. Based on this problem, we propose a solution.

The solution is called an oracle. An oracle connects the off-chain world with smart contracts. Unlike most existing oracles, WINkLink is a decentralized oracle network that provides more secure services than conventional ones.

This paper will discuss our upgrade from FluxAggregator to Off-chain Reporting (OCR) Aggregator, enabling the connection of on-chain smart contracts with off-chain real-world data through WINkLink nodes.

# Contents

# Introduction

Smart contracts are applications deployed and executed on decentralized systems. Once deployed on a blockchain, smart contracts cannot be modified. Compared to traditional contracts, smart contracts offer increased security as all parties, including the author, have equal authority. They are automatically executed when they meet the specified requirements, enabling agreement among all contract participants without the need for trust.

Smart contracts are unable to retrieve off-chain data, such as information obtained through APIs. This limitation arises from the consensus mechanism of the blockchain. To address this issue, we propose the implementation of WINkLink, a decentralized oracle network.

WINkLink serves as a decentralized oracle network that reduces the reliance on trust among contract parties. It ensures the security of the entire smart contract execution process, including the retrieval of data from off-chain sources. This prerequisite is essential for establishing a connection between smart contracts and the real world, ultimately replacing traditional digital contracts.

To facilitate the broader application of smart contracts, ensuring the accuracy of input/output data is crucial. Examples of data requirements for smart contracts include:

- Securities smart contracts, such as bonds and interest rate derivatives, necessitate access to APIs for reporting market prices and market reference data.
- Insurance smart contracts require data feeds concerning IoT data related to the insurable event, such as confirming whether the warehouse's magnetic door was locked during a breach or verifying the online status of the company's firewall.
- Trade finance smart contracts rely on GPS data for shipments, data from supply chain ERP systems, and customs data regarding the shipped goods, all of which confirm compliance with contractual obligations.

Payment messages typically need to be transmitted to off-chain institutions, such as the bank system. WINkLink securely outputs data to off-chain systems, facilitating the connection to the real world and ensuring the tamper-proof nature of smart contracts.

# WINkLink System Overview

WINkLink aims to bridge the gap between the on-chain and off-chain worlds, starting with its deployment on the TRON network and with plans to support other blockchain networks in the future. The development of WINkLink adheres to a modular approach, which enables easy implementation of future optimizations and enhancements.

## Oracle Node

Like the blockchain, the oracle network comprises many nodes. Each node has its own data source set, which may overlap with the others'. An oracle aggregates data from its data sources and sends the aggregated result to the request. To ensure accuracy, a request may select multiple nodes. As faulty nodes may exist, it is important to have a plan in place to mitigate their influence.

To ensure the robustness of the network, WINkLink oracle nodes adhere to the principles of Byzantine fault tolerance (BFT), forming a peer-to-peer (P2P) network. Within the P2P network, individual oracles exchange messages with each other over a network and are identified by their network endpoints, namely a certificate on their cryptographic key material, enabling them to authenticate with each other. In BFT, $2f + 1$ represents the minimum number of honest nodes or processes required to tolerate up to f faulty or malicious nodes or processes. The value of f represents the maximum number of faults the system can withstand while preserving its integrity.

By mandating the presence of $2f + 1$ honest nodes, the system guarantees a sufficient number of correct nodes to outvote the faulty ones during the consensus process. This approach ensures the network's ability to reach consensus and make dependable decisions, even in the face of Byzantine faults.

The choice of $2f + 1$ is crucial as it represents the minimum threshold for achieving Byzantine fault tolerance. This balance strikes a balance between system resilience and efficiency, allowing the network to withstand a specific number of faults while upholding the overall trustworthiness and consistency of the shared or processed data.

## Design Philosophy

The design has multiple key goals. It aims to:

- Ensure the protocol's resilience against various failures caused by malicious actors or software bugs. A security model has been implemented to limit the number of faulty oracles without restricting the types of faults.

- Create a simple and easily implementable design that can be quickly deployed to meet market demands. Choices are made based on simplicity to reduce defects in the system. For instance, on-chain communication is simplified by using Trongrid HTTP and gRPC endpoints instead of hosting a full node.

- Minimize transaction fees by leveraging the virtually free communication between oracles and off-chain computation, while considering the involvement of Tron transactions when communicating with a specific entity (referred to as "*C*"). Despite Tron transactions having low gas fees, efforts are made to maintain low overall transaction fees, even if it necessitates more off-chain computation and networking resources

- Reduce latency as much as possible to minimize the time between initiating the signing protocol and including the transaction on the blockchain. This is important for providing up-to-date data to DeFi trading platforms. The protocol's performance is primarily limited by network transmission latency, which is minimal due to the small amount of transmitted data. The aim is to achieve a report generation time within a few seconds, including the time needed to transmit the report to "*C*."

The development was broken down into two main areas: Off-Chain Report Computation and On-Chain Report Transmission.

## Off-chain Computation

Within the Off-chain Computation, we can further break it down into two components that manage report creation. Namely:

1. Pacemaker
2. Report Generator

### Pacemaker

The oracle report generation for contract *C* is facilitated by the pacemaker protocol, which divides the process into epochs with designated leaders. While it does not guarantee consensus, the protocol relies on contract *C* to resolve any ambiguities during epoch transitions. The pacemaker protocol continuously runs and periodically initiates new epochs and corresponding report generation instances. It monitors progress through events and can abort the current instance if insufficient progress is observed. The pacemaker protocol does not directly send reports to *C* but instead hands them off to the transmission protocol. Additionally, it responds to change of leader events, signaling the end of an epoch and allowing for the initiation of the next epoch with potentially new instances and leaders.

### Report Generator

The Report Generation process is divided into epochs, with each epoch having a unique identifier and a leader node. The protocol operates in rounds within each epoch, where observations are collected. From the observations, a median value is reported, and a signed oracle report is generated. The report is then handed over to the transmission protocol for delivery to *C*, provided certain conditions are met.

By selecting the median from a set of more than $2f$ observations, the protocol guarantees that the reported value is reasonable since Byzantine oracles cannot manipulate it beyond the range of observations submitted by the honest oracles.

The frequency of rounds and observation gathering is controlled by the leader and a timeout value. The timeout value must be smaller than the progress timeout and larger than the network latency required to complete a full iteration of the report generation protocol, with an additional safety margin.

Once a sufficient number of observations are gathered, the report is transmitted to *C* through the transmission protocol, including enough oracle signatures for verification. However, the leader and oracles only generate and participate in producing the report if there is a significant change in the data-stream value or a specific time interval has passed since the last report by *C*. This helps prevent unnecessary reports.

## On-chain Transmission

### Transmission

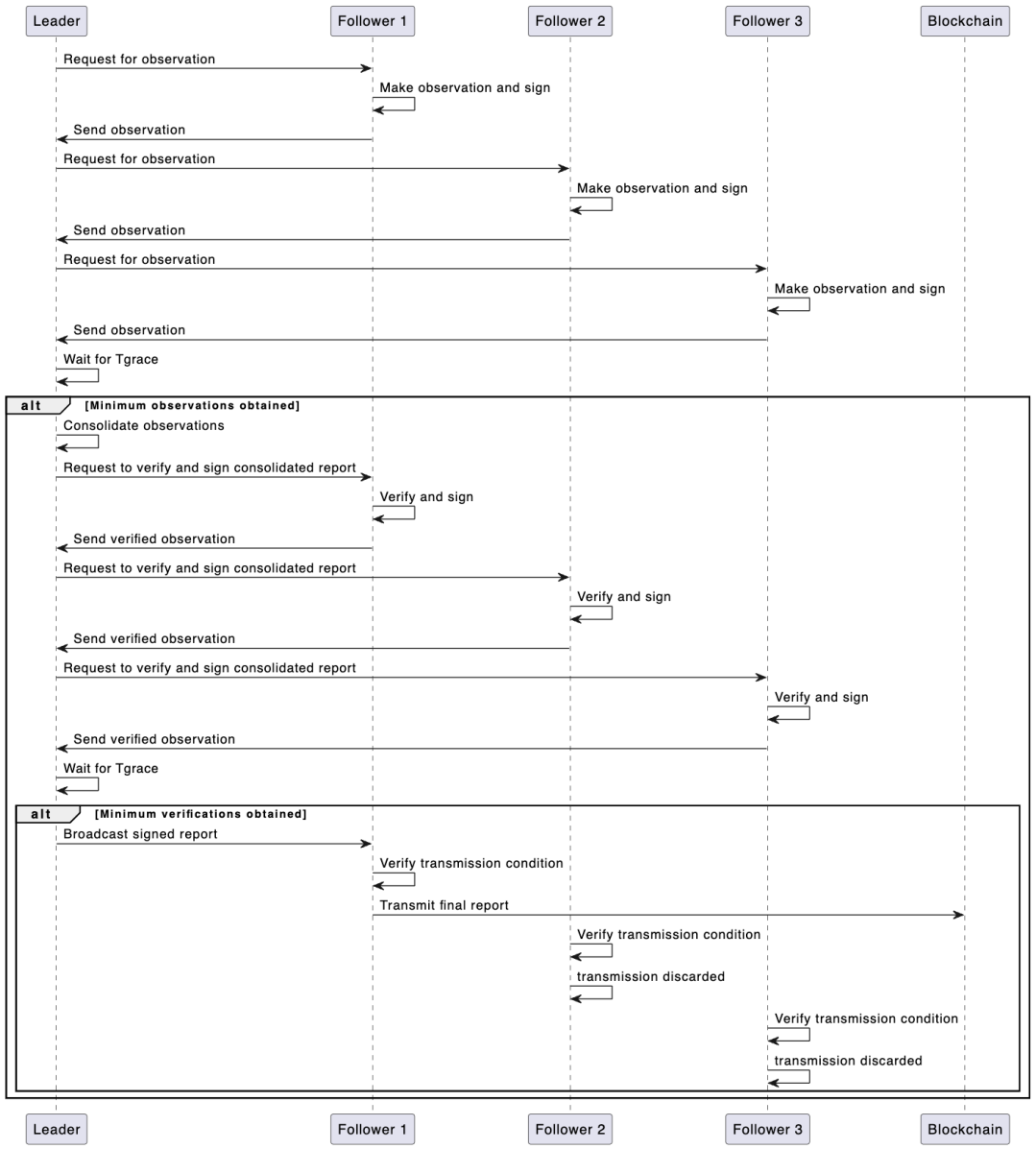
Finally, after *Report R* is generated, the report generation algorithm initiates the transmission protocol simultaneously across all oracles, assuming ideal conditions.

To minimize gas expenses and avoid unnecessary transmissions, the algorithm incorporates a filtering mechanism for incoming reports. This is particularly valuable when multiple rounds yield similar reports in quick succession. Only the initial report requires transmission, while subsequent reports are disregarded.

The algorithm maintains a record of the most recent incoming report (referred to as *L*). To pass through the filter, a report (designated as *O*) must meet either of two conditions. Firstly, if *C* has already received a report that is at least as recent as L, indicating the absence of a report backlog. Secondly, if *O* demonstrates a significant deviation in the median observation value compared to the median value in *L*.

Once the filter conditions are met, a random node from the oracle network is chosen as the transmitter to submit the report to the blockchain. This approach significantly reduces gas consumption since only one transaction is needed, even though multiple nodes have contributed their responses.

# WINkLink OCR workflow



# An ideally secure oracle

An instructive and principled way to reason about oracle security stems from the following thought experiment. We consider a hypothetical scenario where there exists a trusted third party capable of consistently performing instructions honestly. This idealized oracle, referred to as an ideal oracle, obtains data from a trustworthy data source and ensures its security through the following tasks:

- Accept request: The ideal oracle ingests a request $Req = (Src, \tau, q)$ from a smart contract *USER-SC*, which specifies a target data source $Src$, a time or range of times $\tau$, and a query $q$.
- Obtain data: The ideal oracle sends the query $q$ to the specified data source $Src$ at the designated time $\tau$.
- Return data: Upon receiving the answer $a$, the ideal oracle returns it to the smart contract.

The ideal oracle plays a crucial role as it serves as a vital bridge between the data source and the USER-SC, providing accurate and timely data.

In many scenarios, the requested data is not suitable for public access. To maintain confidentiality, the ideal oracle ensures that data requests are always kept secure. Requests are encrypted, and the ideal oracle holds the public key necessary for decryption.

An ideal oracle should always be available, without any downtime, and should not deny any requests that come its way.

However, it is important to acknowledge that there is no data source in the world that can be considered 100% trustworthy. Data carries inherent risks of tampering due to various possible factors such as vulnerabilities or cheating on websites. Additionally, it is unrealistic to expect a perfect third party to run an oracle.

While the ideal oracle does not exist in reality, our aim is to make WINkLink come closer to embodying its principles and ideals.

# Data Aggregation and Security

WINkLink proposes two approaches to avoid the appearance of faulty nodes: distribution of data sources and oracles.

## Data Source

We can obtain data from several different data sources to mitigate the impact of an abnormal data source. An aggregating function can aggregate the results into a single output. There can be many ways to do aggregation, such as calculating the weighted average after removing abnormal data.

Data sources may obtain data from each other, and this causes the aggregated result inaccurate. WINkLink will concentrate on solving these problems, and report on the independence of data sources.

## Data Aggregation Benchmarking

Given a 7 oracle node network, FluxAggregator will need to transact 7 times onto the chain to fulfill the price feed. However, OCR will only require 1 transaction to achieve the same goal. This translates to ~110 TRX gas consumption on OCR while a total of ~350 TRX on Flux, resulting in approximately 65% gas cost savings. With more nodes, the cost savings will be even greater as the transaction ratio is 1:$n$, where $n$ is the number of nodes.

## Contract-upgrade service

Once deployed, no one can control the actions of smart contracts, highlighting the importance of oracle security. An oracle providing incorrect data can cause significant losses for a decentralized exchange.

For security reasons, WINkLink proposes a contract-upgrade service. This service will be managed by organizations that launch WINkLink nodes and adhere to WINkLink's decentralized design philosophy.

Numerous smart contract hack events have demonstrated the existence of significant security risks. This is precisely why we propose the contract-upgrade service.

The contract-upgrade service is optional, allowing users to decide whether to activate it based on their needs.

When vulnerabilities are discovered, the contract-upgrade service will deploy a new set of oracle contracts. Both versions of the contracts will coexist and be available for use. With the philosophy of decentralization, users will have a flag to control their selection of requesting contracts from either set.

WINkLink is a decentralized oracle network. The decision to use the new version lies with the user rather than the contract developer. Additionally, we anticipate that providers will be able to support multiple versions of WINkLink-SC developed by the community.

# WIN token usage

WIN is a *TRC-20* token. The WINkLink network utilises the WIN token to pay WINkLink Node operators for the retrieval of data from off-chain data feeds, formatting of data into blockchain readable formats, off-chain computation, and uptime guarantees they provide as operators. WINkLink will power WIN token in many ways.

# Roadmap and Future Plan

For the future roadmap of WINkLink Oracle will be focusing on the following areas:

1. Improving the safety and reliability of the oracle
2. Increasing the variety of quality data source for the oracle ecosystem
3. Improving the ease of use to integrate with WINkLink oracle to serve the mass market use case on oracle

## Validation system

The validation system should monitor the on-chain behavior of oracles, providing objective performance metrics to assist users in their selections. The monitoring process will encompass two perspectives:

1. Availability: This involves tracking oracle failures and providing prompt responses to user queries.
2. Correctness: This entails recording any deviations observed in comparison to other oracle nodes.

WINkLink-SC has the capability to monitor the activities of all oracles. The statistics regarding availability and correctness will be publicly available and published on the blockchain.

## Reputation system

The reputation system aims to record user ratings of oracle service providers and nodes. Reports generated by the validation system can serve as the primary factor in determining reputation. Additionally, other information such as users' familiarity with the brand, operating entities, and architectures of oracles may also be considered. The reputation system can provide reports obtained from other smart contracts. Furthermore, we acknowledge the need to analyze massive amounts of data and thus, consider calculating reputation metrics off-chain.

For a given oracle operator, the reputation system initially proposes support for the following metrics. These metrics can be evaluated at the granularity of specific assignment types as well as in general for all types supported by a node:

1. Total number of assigned requests: This refers to the overall count of past requests, including both fulfilled and unfulfilled ones.
2. Total number of completed requests: This represents the total number of requests that have been fulfilled.
3. Total number of accepted requests: This measures the number of fulfilled requests that were ultimately accepted by the user.
4. Average time to respond: This metric calculates the average response time based on completed requests that were not accepted.
5. Amount of penalty payments: This metric captures the total amount paid by the service provider as penalties.

The reputation system will incentivize oracle service providers to maintain higher availability and performance standards. We hope that the reputation system will serve as a guide for users in selecting nodes and services.

## Certification Service

The oracle node is vulnerable to Sybil attacks, wherein an attacker aims to gain control over the oracle pool by manipulating multiple nodes that appear independent. By providing inaccurate data at specific times, these nodes can influence large transactions in high-value contracts.

To minimize costs, a Sybil attacker may employ a technique known as mirroring, where an oracle node is compelled to retrieve data from a single source but presents it as if it came from multiple sources. Mirroring provides benefits to the adversary regardless of whether they choose to transmit false data.

The Certification Service offers endorsements based on various aspects of oracle deployment and behavior. It will monitor the statistics of the Validation System concerning oracles and conduct post-hoc spot-checking of on-chain answers, particularly for high-value transactions. This involves comparing the responses with answers obtained directly from reputable data sources.

In addition to reputation metrics and automated on-chain/off-chain systems for fraud detection, the Certification Service is designed to identify Sybil attacks and other malicious activities that automated on-chain systems may not detect.

# Conclusion

We introduce WINkLink, a decentralized oracle network that encompasses both on-chain and off-chain components. We outline our commitment to security and decentralization, as well as highlight existing design flaws and our plans for future developments.

Decentralization serves as the fundamental principle of both blockchain technology and WINkLink. We remain steadfast in our dedication to decentralization and strive to enhance the performance and security of the oracle network.

WINkLink is a project that builds upon the achievements of pioneers in the field. We highly value our community and will continue to develop WINkLink in an open-source manner. We greatly appreciate any reviews and suggestions from the community, as we believe that collaboration will propel the advancement of blockchain and smart contracts.