WINkLink

A decentralised oracle network on TRON

6 October 2020 (v1.0)

Abstract. Smart contract is one of the most important parts of modern blockchains. Smart contracts are deployed on blockchains, triggered automatically and cannot be modified after deployed. 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 to this.

The solution is called an *oracle.* An oracle connects the off-chain world with smart contracts. Differ from most existing oracles, WINkLink is a decentralised oracle network, which provides more secure services than the normal ones.

This paper details the on-chain components for smart contracts connecting with the off-chain world and the underlying modules of WINkLink nodes. Probable optimisations are also included in this paper, which illustrates the directions for WINkLink in the future.

Contents

Introduction

Smart contracts are applications deployed and executed on decentralised systems. Any changes cannot be made once a smart contract is deployed on a blockchain. Smart contracts are more secure compared with traditional contracts, as anyone(including the author) has the same authority. Smart contracts are automatically executed when they meet the requirements; all parties of the contracts can reach an agreement without trust.

Smart contracts cannot obtain the off-chain data themselves, such as API data. This is caused by the consensus mechanism of the blockchain, and we propose an oracle network, WINkLink, to solve this problem.

WINkLink is a decentralised oracle network. Decentralisation reduces the needs of trust among parties of the contracts, and WINkLink ensures the security of the entire procedure of smart contracts execution, including obtaining data from off-chain sources. This is the prerequisite of connecting smart contracts to the real world and taking the place of traditional digital contracts.

To apply smart contracts to a wider range of scenarios, the accuracy of the input/output data is required. Example data requirement of smart contracts can be as following:

- Securities smart contracts such as bonds, interest rate derivatives, and many others will require access to APIs reporting market prices and market reference data, e.g. interest rates.

- Insurance smart contracts will need data feeds about IoT data related to the insurable event in question, e.g.: was the warehouse's magnetic door locked at the time of the breach, was the company's firewall online or did the flight you had insurance for arriving on time.

- Trade finance smart contracts will need GPS data about shipments, data from supply chain ERP systems, and customs data about the goods being shipped to confirm the fulfilment of contractual obligations.

Payment messages usually should be passed to off-chain institutions(e.g. the bank system). WINkLink can output data securely to off-chain systems, which implements the connection to the real world and ensure the temper-proof of smart contracts.

WINkLink System Overview

WINkLink aims to link the on-chain and off-chain worlds. WINkLink is initially deployed on the TRON network and will support other blockchain networks.

We develop WINkLink with the modularisation concept. It will be easy for us to apply optimizations in the future.

On-chain

Data requests initiated by smart contracts are called *requesting contracts* and denote by USER-SC. The on-chain interface WINkLink interacts with requesting contracts is itself a smart contract which is denoted by WINkLink-SC.

WINkLink has an on-chain module called *aggregator contract*. Users choose nodes and services from a front-end page for the aggregator contract, and then it will calculate the final result for requesting contracts.

Oracle Selection

Oracle services purchasers evaluate their specific requirements, then select nodes and services from the list page. Nodes related data is available in the list for consumers to choose appropriate nodes and services.

We have also considered manual matching is not possible for all situations. In the future, we will design an automated matching mechanism to meet more needs.

Data Aggregation

An aggregator contract collects results from all specified nodes and calculates a result. This result is finally sent to USER-SC.

There is not a universal aggregator contract, for every demand can be different(e.g. return type). WINkLink will include a standard(i.e. a template) for users to customise their contracts.

Off-chain

The off-chain component of WINkLink is the oracle node. Nodes obtain off-chain data separately and finally calculate a single result in the aggregator contract. The following of this paper elaborates on how to aggregate the several responses and return it to USER-SC. The WINkLink nodes are powered by the standard open-source core implementation, which handles standard blockchain interactions, scheduling, and connecting with common external resources.

WINkLink Core

The core software of a node is responsible for interacting with the blockchain, assignment scheduling and work balancing. Work done by WINkLink is called an *assignment.* Every assignment can be divided into *subtasks.* Each subtask passes its result to the next subtask; they run tandemly to get the final result.

WINkLink has several built-in subtasks, including HTTP requests, JSON parsing and conversion to various blockchain formats.

External Adapters

Users can customise subtasks within an *external adapter.* Adapters are external services with a minimal REST API. By adding an intermediate API in front of the program, programs in any programming language can be easily implemented.

Subtask Schemas

With the application of WINkLink becoming wider, there can be many open-source adapters. Any community members can review the code of adapters. It is an essential task to keep the compatibility between adapters because there can be various adapters made by the community.

WINkLink specifies the standard of input/output and formatting.

WINkLink workflow

- USER-SC makes an on-chain request
- WINkLink-SC logs an event for the oracles
- WINkLink core picks up the event and routes the assignment to an adapter

- WINkLink adapter performs a request to an external API
- WINkLink adapter processes the response and passes it back to the core
- WINkLink core reports the data to WINkLink-SC
- WINkLink-SC aggregates responses and gives them back as a single response to USER-SC.

An ideally secure oracle

An instructive, principled way to reason about oracle security stems from the following thought experiment. Assuming there is a trusted third party that can always perform instructions honestly, the oracle runs by it is called an *idea oracle*. The idea oracle obtains data from a trustful data source, to keep secure, it will carry out the following tasks:

- Accept request: Ingest from a smart contract USER-SC a request Req = (Src, τ, q) that specifies a target data source Src, time or range of times τ, and a query q;
- Obtain data: Send query q to Src at time τ ;
- Return data: On receiving answer a, return a to the smart contract.

The idea oracle builds a vital bridge between the data source and USER-SC; it gives accurate data on time.

The requested data is not suitable for the public in many scenarios. The idea oracle will always keep data requests confidential. Requests will be encrypted, and the idea oracle holds the public key.

An idea oracle should always be available, never downtime, and will not deny any requests.

However, there is not a 100% trustworthy data source in the world. The data has risks of tempering due to many possible reasons(e.g. vulnerability or cheating of the website). Also, there cannot be a perfect third party to run an oracle.

The idea oracle does not exist. What we are trying is to make WINkLink closer to it.

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.

Oracle Node

Like the blockchain, many nodes form the oracle network. Each node has its 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. A request may choose several nodes to ensure accuracy. As faulty nodes may exist, there should be a plan to mitigate the influence.

The simplest approach is aggregating in WINkLink-SC. The code is open-source that can be easily verified. Also, any behaviour of WINkLink-SC is on the blockchain, which is completely visible to users. The aggregating function can be the majority function, averaging, Etc.

This simple approach raises a new problem of freeloading. It's easy for a cheating node to copy the data from an honest node. Thus, the cheating node will never have to pay the data sources. This may affect the independence of each result.

Therefore, we introduced a mechanism of commit/reveal. Simply put, the encrypted data will only be decrypted if WINkLink-SC receives enough results. We believe this can reduce cheating nodes to a certain extent.

Due to the high throughput and low transaction fee on the TRON network, currently, we do not have to aggregate the result in off-chain environments to reduce cost.

Contract-upgrade service

No one can control the actions of smart contracts once deployed; this makes the security of the oracle important. A decentralised exchange can suffer a massive loss if it receives incorrect data from an oracle.

WINkLink proposes a contract-upgrade service for security reasons. The service will be run by the organisations who launch WINkLink nodes and follow WINkLink's philosophy of decentralised design.

Many smart contract hack events show there remain significant security risks. This is precisely the reason for our proposing the contract-upgrade service.

Contract-upgrade service is non-mandatory. Users decide whether to turn this on according to their demand.

Contract-upgrade service will deploy a new set of oracle contracts once vulnerabilities are discovered. The two versions of contracts will both exist and available to use. With the philosophy of decentralisation, there will be a flag for users to control. The flag enables requesting contracts to select which set of contracts they would like.

WINkLink is a decentralised oracle network. The choice of whether to use the new version is left to the user but not the contract developer. Also, we expect 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

We'll concentrate mainly on the direction of improving safety and reliability.

Validation system

The validation system should monitor on-chain oracle behaviour, providing objective performance metric to guide users' selections. The monitor will be in two perspectives:

- Availability: recording oracle failure and responding to queries.
- Correctness: recording the deviation compared with other oracle nodes.

WINkLink-SC can monitor the activities of all oracles. The statistics of availability and correctness will be published on the blockchain.

Reputation system

The reputation system is to record user ratings of oracle service providers and nodes. Reports generated by the validation system can be the primary factor of the reputation. Information other from this can include users' familiarity with oracles' brands, operating entities, and architectures.

The reputation system can provide reports obtained by other smart contracts. We also consider calculating the reputation metrics off-chain, as there can be massive data to be analyzed.

For a given oracle operator, the reputation system is initially proposed as supporting the following metrics, both at the granularity of specific assignment types, and also in general for all types supported by a node:

- Total number of assigned requests: the total number of past requests, including fulfilled and unfulfilled requests

- Total number of completed requests: the total number of all fulfilled requests

- Total number of accepted requests: the number of fulfilled requests which finally accepted by the user

- Average time to respond: the average time is calculated based on complete(not accepted) requests

- Amount of penalty payments: the total amount paid by the service provider

The reputation system will encourage oracle service providers to keep higher availability and performance. We hope that the reputation system will become a vane for users to choose nodes and services.

Certification Service

The oracle node is exposed to the risk of Sybil attacks. An attacker attempts to dominate the oracle pool by controlling a number of nodes that seem independent. These nodes can provide wrong data in any specific time to influence large transactions in high-value contracts.

To reduce the cost, a Sybil attacker can adopt a behaviour called mirroring, force an oracle node to obtain data from a single data source, and disguise it as it was from multiple sources. Mirroring benefits an adversary whether or not it chooses to send false data.

The Certification Service supports endorsements based on several features of oracle deployment and behaviour. It would monitor the Validation System statistics on oracles and perform post-hoc spot-checking of on-chain answers—particularly for high-value transactions—comparing them with answers obtained directly from reputable data sources.

In addition to the reputation metrics, automated on-chain and automated off-chain systems for fraud detection, the Certification Service is planned as a means to identify Sybil attacks and other malfeasance that automated on-chain systems cannot.

Conclusion

We introduce the decentralised oracle network WINkLink, including its on and off-chain components. We interpret our schemes of security and decentralisation. We also discovered the existing design flaws and give plans for future developments.

Decentralisation is the foundation of the blockchain, and of course, WINkLink. We will always adhere to the concept of decentralisation, further improve the performance and security of the oracle network.

WINkLink is a project that stands on the shoulders of giants. We will always value the community and continue to develop WINkLink in an open-source manner. We are appreciated for any reviews and suggestions from the community. We hope WINkLink can promote the development of blockchain and smart contracts.