Blockchain Oracles

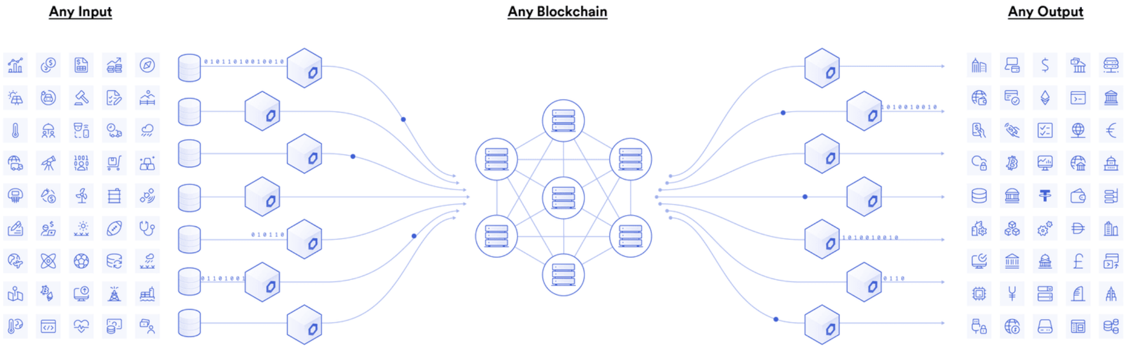
A short introduction for the *Blockchain Technology* Seminar SS22

7/12/2022

Outline

- **Motivation**
- 🌳 **Oracle Taxonomy**
- **Centralized Oracle: Provable**
- **Decentralized Oracle: ⬡ Chain Link**
  - Architecture 🏗️
  - Solutions
    - Data Feeds 💾
    - VRF - random number generator 🎲
    - ...
  - Monetizing Data 🤑
- Case Study: Arbol 🌾
- 🕳️ 🐇 **Enter the rabbit hole**

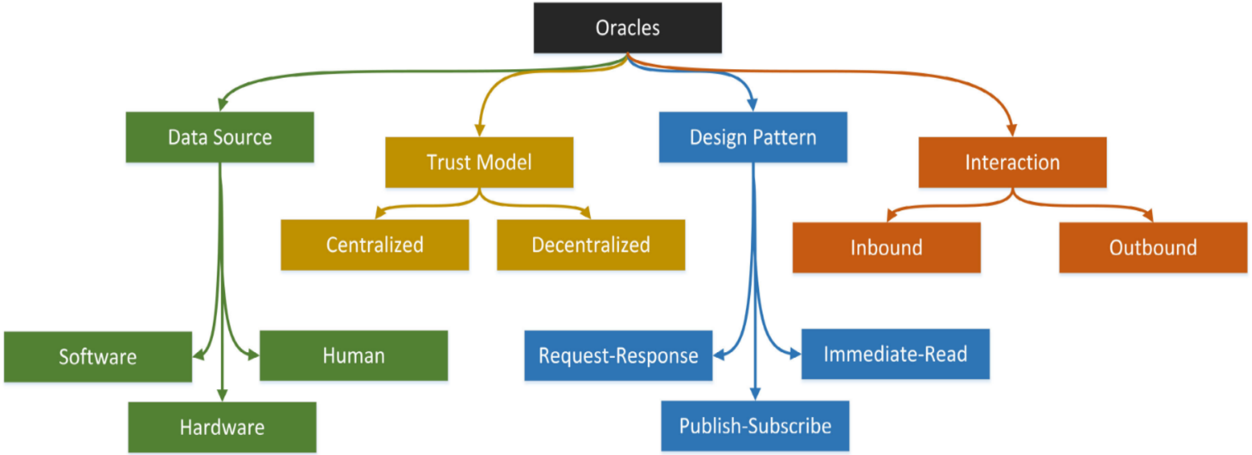
Motivation



Source

- dApps need to be able to interact with real-world events
- enable blockchain systems to access existing data sources, legacy systems and advanced computations
- enable *any blockchain* to create *any output* with **any input**
- **BASICALLY: We need to get data on the chain** ⛓

🌳 Oracle Taxonomy



Source

Categorize the following example 📥

An oracle supplies a smart contract with the information, whether a package was dropped of on the RFID sensor in front of our door 📦

🌳 Data Source

- **Software Oracles**
  - deal with data originating from the internet
  - *e.g. asset prices, currency exchange rates,…*
- **Hardware Oracles**
  - gather data **directly** from the physical source
  - *e.g. scanners, RFID chips, temperature sensors,…*
- **Human Oracles**
  - rely on people’s actions
  - *e.g. outcome of a soccer match, vote on the best contestant,…*

🌳 Interaction

with the external world is either...

- inserting data **to** the blockchain
- delivering data **from** the blockchain

Inbound Oracles

- **to**
- *e.g. asset price which can then be automatically purchased by smart-contract*

Outbound Oracles

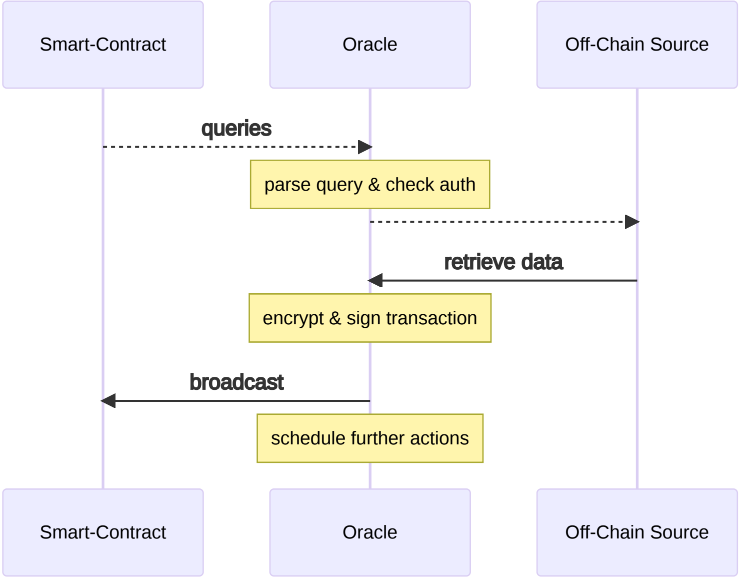
- **from**
- *e.g. smart-lock in AirBnB is opened once ETH payment arrives on smart-contract address*

🌳 Design Pattern

Request-response

- data space to huge to be stored in smart-contract
- users only need small data subset at a time
- off-chain infrastructure monitors on-chain smart-contract calls
- common in client-server architectures
- allows for two-way communication

e.g. synchronize interest rate of a smart bond daily



Smart-Contract

Oracle

Off-Chain Source

queries

parse query & check auth

retrieve data

encrypt & sign transaction

broadcast

schedule further actions

Smart-Contract

Oracle

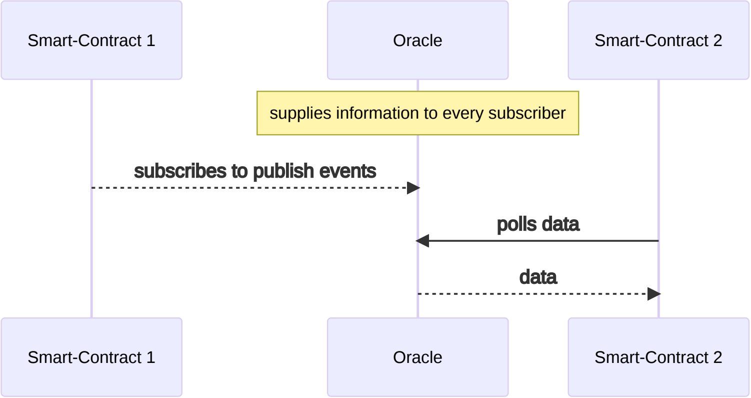
Off-Chain Source

🌳 Design Pattern

Publish-subscribe

- effectively provides data broadcast service
  - *think of a RSS feed*
  - data is expected to change
- subscribers can either
  - poll for information with smart-contract
  - listen for changes via off-chain daemon

e.g. average temperature in Germany

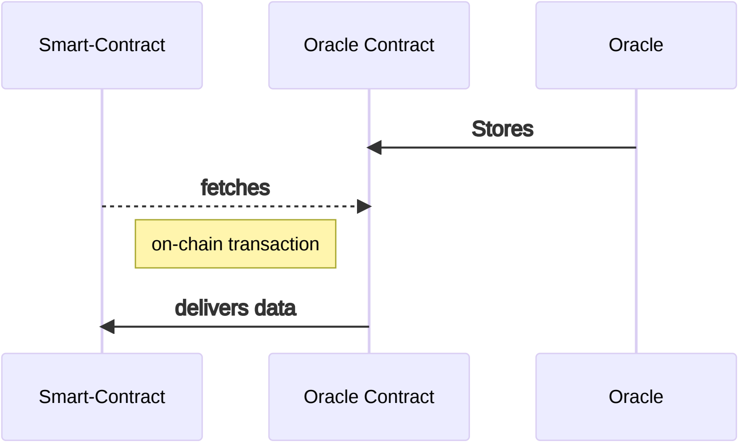


🌳 Design Pattern

Immediate-read

- provide data necessary for immediate decision
  - *is this student enrolled?*
- most of the times queried in a *JIT* manner
- attractive to companies that would otherwise need to supply their own infrastructure
- often stored in contract storage
  - stored on chain
  - less gas fee intensive

e.g. an oracle to for certificates of past academic achievements



🌳 Trust Model

Centralized

- high efficiency
- single point of failure regarding
  - availability
  - accessibility
  - certainty about information validity
- corrupted oracle could…
  - manipulate on-chain data
  - break consensus
  - attack the network
- 🚫 **defeats the whole purpose of a decentralized blockchain application**

Decentralized

- prevents
  - data manipulation
  - inaccuracy
  - downtime
- i.g. tries to avoid counter-party risk
- can be referred to as consensus oracles
- ✅ **this is what we generally want to use in an DLT context**

Centralized Oracle: Provable

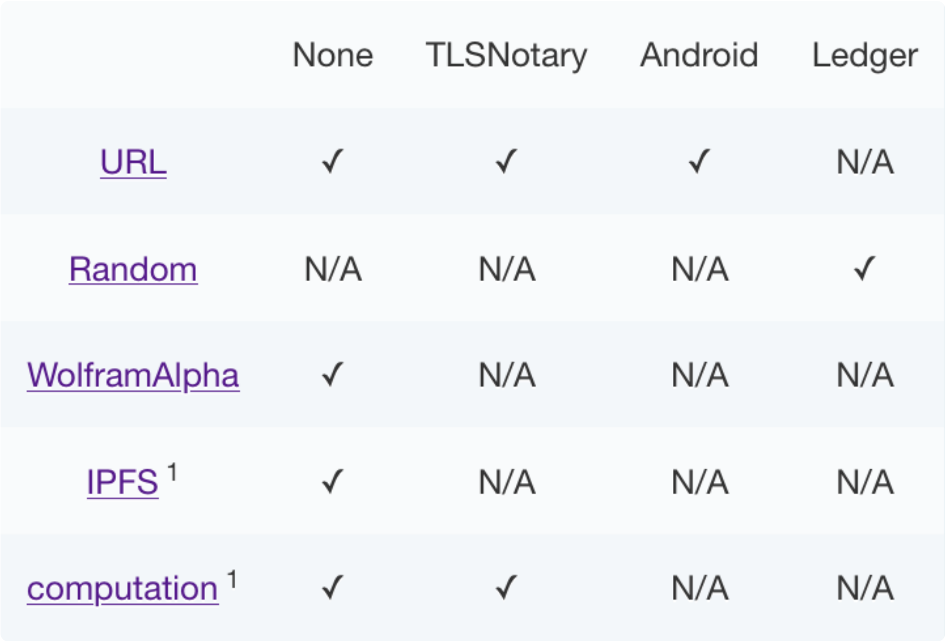
- easy to use via simple queries
- blockchain agnostic
  - most services live off-chain
  - designed for a blockchain context
  - *Provable HTTP API is also provided*
- military-grade security
  - multiple types of **authenticity proofs**: software & hardware based
  - ensure delivery of untampered data
- 🚨 *"Most of the software we produce is open-source and all the critical pieces are published as such."*
- certified
  - entire external audit trail is published
  - as of now the link is broken 🤔
- flexible & efficient

📦 Provable Query Example

```
pragma solidity ^0.4.22;
import "github.com/provable-things/ethereum-api/provableAPI_0.4.25.sol";

contract ExampleContract is usingProvable {

  // rest of contract omitted for brevity...

   function updatePrice() payable {
       if (provable_getPrice("URL") > this.balance) {
           LogNewProvableQuery("Provable query was NOT sent, please add some ETH to cover for the query fee");
       } else {
           LogNewProvableQuery("Provable query was sent, standing by for the answer..");
           provable_query("URL", "json(https://api.pro.coinbase.com/products/ETH-USD/ticker).price");
       }
   }
}
```

Request: `(<data source type>, <query>, <optional: authenticity proof type>)`

Data Sources & Authenticity Proof Types



	None	TLSNotary	Android	Ledger
URL	✓	✓	✓	N/A
Random	N/A	N/A	N/A	✓
WolframAlpha	✓	N/A	N/A	N/A
IPFS [1]	✓	N/A	N/A	N/A
computation [1]	✓	✓	N/A	N/A

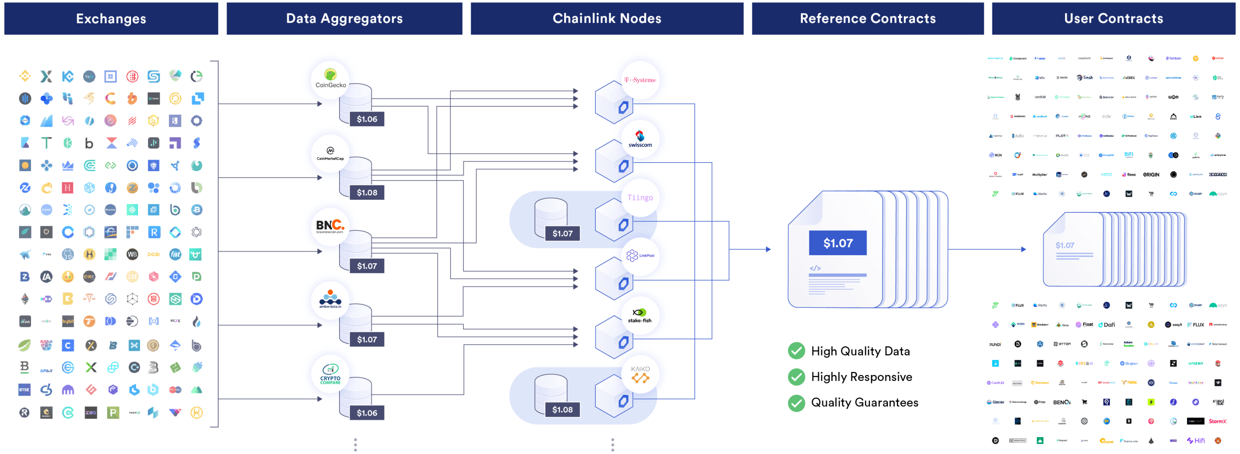
Source

Provable's Downsides

- limited EVM functionality
- inefficient handling of
  - opcodes
  - precompiles
  - precision bound floats
- high gas costs
- absence of confidentiality & privacy...
- ...

In addition to Provable's centralized infrastructure this does sound suboptimal 🤔

Chainlink



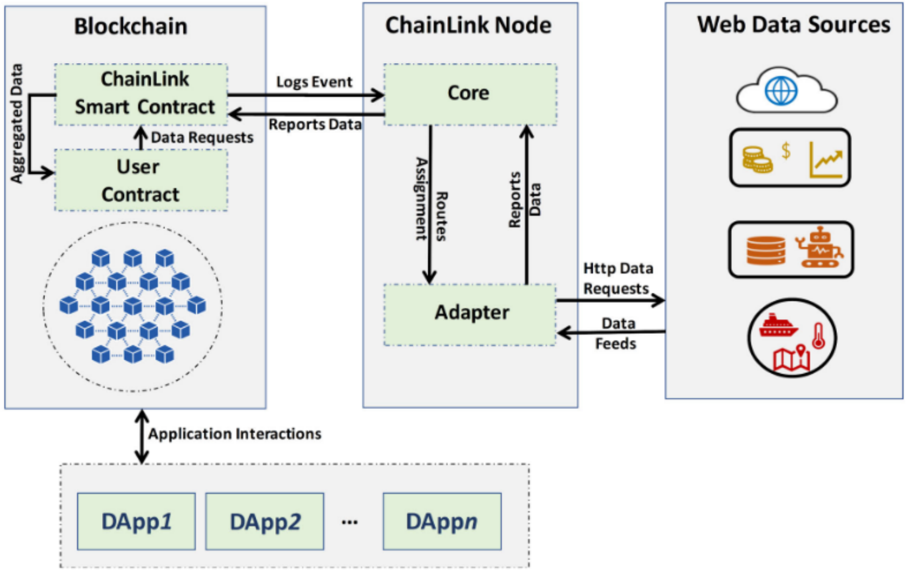
- ✓ High Quality Data
- ✓ Highly Responsive
- ✓ Quality Guarantees

Source

Introduction ✨

> Chainlink is a Decentralized Oracle Network (DON) aimed at enhancing and extending the capabilities of smart-contracts on a given main chain.

- DON’s serve as a flexible and powerful tool for dApp developers
- provide high quality data due to consensus mechanism
- necessary for up to 90% of potential use cases of dApps

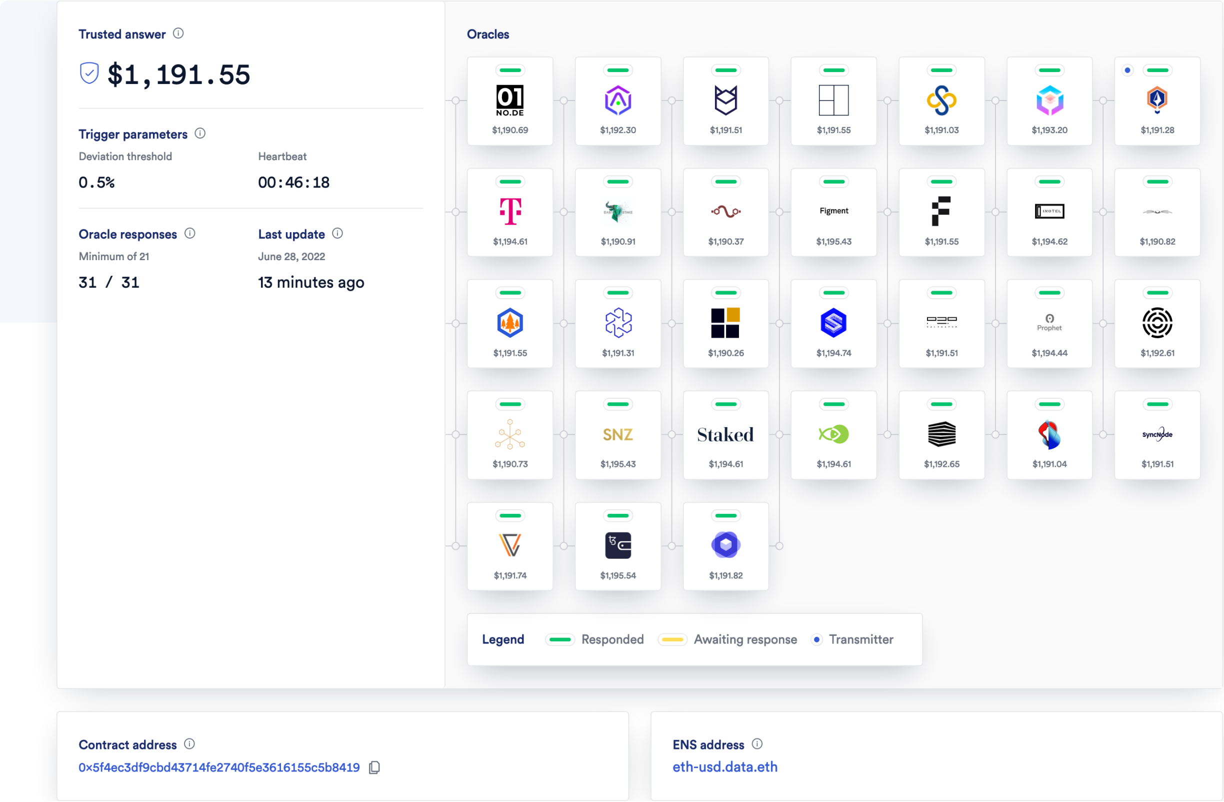


Introduction ✨

Chainlink offers multiple products…

- **Data Feeds**
- **VRF**
  - verifiable, tamperproof
  - low cost
  - random number generator
- **Keeper**
- **Proof of Reserve**
- **Cross-Chain Communication**

Introduction ✨



Trusted answer ⓘ

$1,191.55

Trigger parameters ⓘ

Deviation threshold: 0.5%

Heartbeat: 00:46:18

Oracle responses ⓘ

Minimum of 21

31 / 31

Last update ⓘ

June 28, 2022

13 minutes ago

Oracles

01 NO.DE $1,190.69	$1,192.30	$1,191.51	$1,191.55	$1,191.03	$1,193.20	● $1,191.28
$1,194.61	$1,190.91	$1,190.37	Figment $1,195.43	$1,191.55	$1,194.62	$1,190.82
$1,191.55	$1,191.31	$1,190.26	$1,194.74	$1,191.51	Prophet $1,194.44	$1,192.61
$1,190.73	SNZ $1,195.43	Staked $1,194.61	$1,194.61	$1,192.65	$1,191.04	SyncNode $1,191.51
$1,191.74	$1,195.54	$1,191.82				

Legend 🟩 Responded 🟨 Awaiting response ● Transmitter

Contract address ⓘ

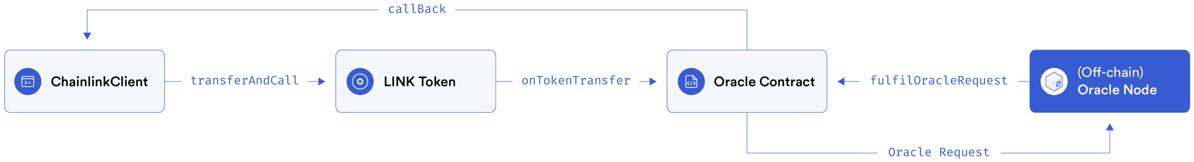
0x5f4ec3df9cbd43714fe2740f5e3616155c5b8419

ENS address ⓘ

eth-usd.data.eth

Source

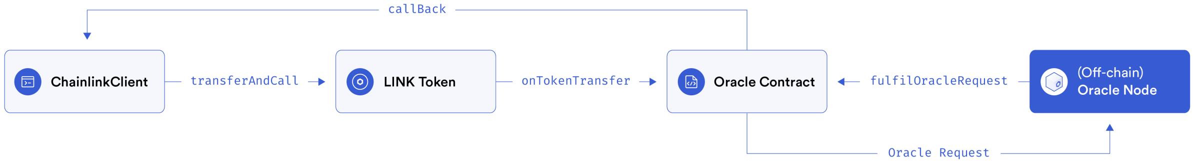
Request Model 🏗️



Source

- **ChainlinkClient** available in the smart contract library
- **LINK Token**
  - used to **compensate node operators**
  - `ERC 667` compliant src
- **Oracle Contract**
- **Oracle Node** (*Off-chain*)

Request Model 🏗️



callBack

ChainlinkClient

transferAndCall

LINK Token

onTokenTransfer

Oracle Contract

fulfilOracleRequest

(Off-chain) Oracle Node

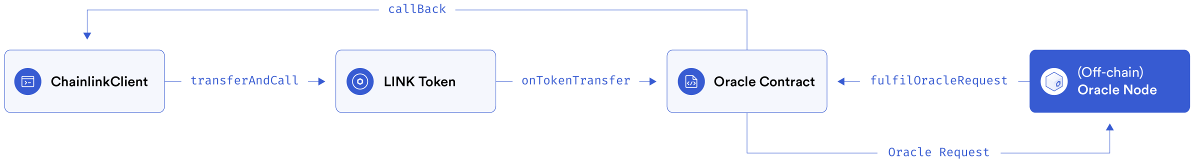
Oracle Request

Source

Oracle Contract

- is contacted if sufficient LINK is available
- owned by node operators
- `Request`
  - `oracle address`, `job ID` & `callback function`
- `Fulfillment`
  - `fulfillOracleRequest` function returns result of request to specified callback

Request Model 🏗️



Source

Oracle Node

- listens to events emitted by the corresponding smart contract: `OracleRequest` event
- creates request and converts result into blockchain compatible data

if you want to run a node 😎

Data Model 🏗️

- **Data Aggregation** *example SOL contract*
  - feed is created by multiple independent operators
  - further enhanced by Off-Chain Reporting - *we talk about that later* ⏰
- **Shared Data Resource**
  - data feeds are built & funded by the users relying on the data
- **Decentralized Oracle Network (DON)**
  - data feeds are updated by an decentralized oracle Network
  - oracles are rewarded for publishing data
  - feeds are only updated if a minimum number of responses are returned
  - data is published during a an aggregation round

DON Components

three contracts…

- **consumer**
  - use data feed

```
AggregatorV3Interface feed = AggregatorV3Interface(address);
return feed.latestRoundData();
```

- **proxy**
  - on-chain
  - enable upgrades/changes of underlying aggregator w/o breaking on-chain functionality
- **aggregator**
  - receives periodic updates from oracle network
  - *if … triggered*
    - deviation threshold
    - heartbeat threshold

Off-Chain Reporting 🏗️

Design Goals

- Resilience
- Simplicity
- Low transaction fees
- Low latency

Simple Analogy :

> Ship an order of multiple items from an online store in one package instead of multiple.

Functionality :

- nodes communicate through a P2P network
- lightweight conensus algorithm decides on which data is included
- aggregated transaction is transmitted
- new *"node leader"* is regularly elected

Data Feeds 💾

are…

- easy to add via `chainlink` npm package
- smart contracts consuming data feeds can be written in …
  - Solidity
  - web3.js / ether.js
  - Web3.py / Vyper
  - …
- some examples of **available data feeds** on Ethereum
  - `AAPL / USD`: Contract
  - `BTC / ETH`: Contract
- migrated to ENS

Feeds are currently available on:

- Ethereum
- BNB
- Polygon (Matic)
- HECO
- Gnosis (xDai)
- Avalanche
- Fantom
- Arbitrum
- Harmony
- Optimism
- Moonrive
- *Solana*

Solidity Example 💾

```
pragma solidity ^0.8.7;

import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";

contract PriceConsumerV3 {

    AggregatorV3Interface internal priceFeed;

    constructor() {
        priceFeed = AggregatorV3Interface(0x8A753747A1Fa494EC906cE90E9f37563A8AF630e);
    }

   function getLatestPrice() public view returns (int) {
        (
            /*uint80 roundID*/,
            int price,
            /*uint startedAt*/,
            /*uint timeStamp*/,
            /*uint80 answeredInRound*/
        ) = priceFeed.latestRoundData();
        return price;
    }
}
```

Python Example 💾

```python
web3 = Web3(Web3.HTTPProvider('https://rinkeby.infura.io/v3/<infura_project_id>'))
abi = '...'
addr = '0x8A753747A1Fa494EC906cE90E9f37563A8AF630e'
contract = web3.eth.contract(address=addr, abi=abi)
latestData = contract.functions.latestRoundData().call()
# latestData now holds the latest ETH/USD price
```

- the API is quite simple
- everyone should find a language to utilize it with

VRF - random number generator 🎲

Why is randomness a problem? 🤔

used for...

- **NFTs**: generation of attributes
- **Gaming**: matchmaking, critical hits, draw order, random events,...
- **Process Ordering**: public sales, auctions,...
- **Entity Selection**: random picker

optimally the generated numbers would be...

- actually random 😄 (as close as possible)
- verifiable via cryptographic proof
- tamper proof
- scalable & cheap (if you are a dev 👨🏻‍💻)

VRF 🎲

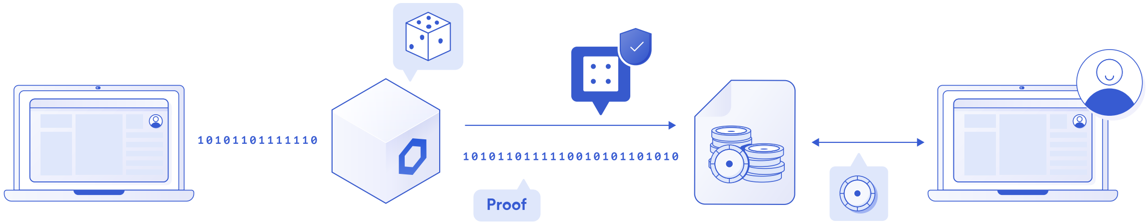
How does we use it?

```
unit256 public randomResult;
function fulfillRandomness(uint256 requestId, unit256[] randomness) internal override {
      randomResult = (randomness[0] % 50) + 1;
}
```

Fulfilling request isn’t free:

- gas price
- callback gas
- verification gas
- gas lane
- callback gas limit

VRF 🎲



Source

- on-chain block data is used as input
- random results is verified on-chain **before** it can be consumed

⇢ this is an advantagous paradigm even for off-chain applications

Keepers ⚙

> Decentralized smart contract automation

Use Cases

- harvest yield
- automated trading
- trigger asset distribution
- liquidations
- ...

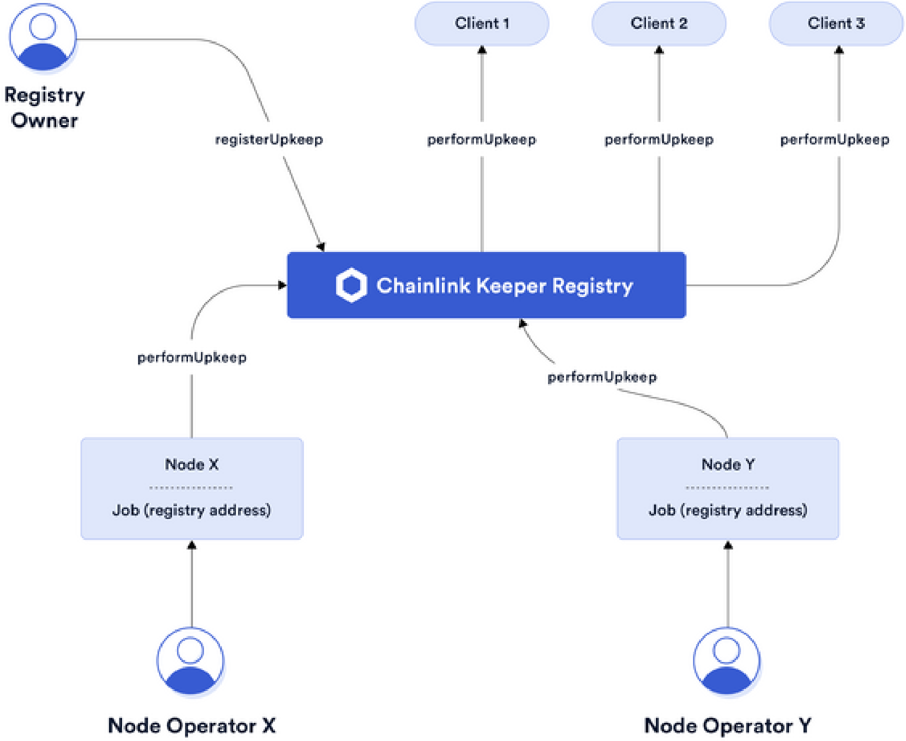
Currently supported on

- Ethereum
- Polygon (Matic)
- BNB
- Avalanche
- Fantom

Architecture ⚙

- **Upkeeps**
  - outsourced maintenance tasks
  - must be *Keepers-compatible*
- **Keeper registry**: contract that is used to register & manage Upkeeps
- **Keepers**: Network nodes

Upkeeps must be sufficiently funded using LINK 💰



Creating your Upkeep ⚙

Choose your trigger (not in a Twitter users way 😄...)

- **time-based**
  - scheduled using CRON
- **custom logic**
  - defined in custom smart contract

Remember that we used to send funds with our requests?

- Upkeeps are funded using the registry

Summary

- Keepers provide a form of decentralized DevOps
- allow for the reduction of gas fees due to off-chain computations
  - several protocols outsourced their maintenance tasks to Keepers
- enables gas fee prediction due to the possibility to set gas fee limits

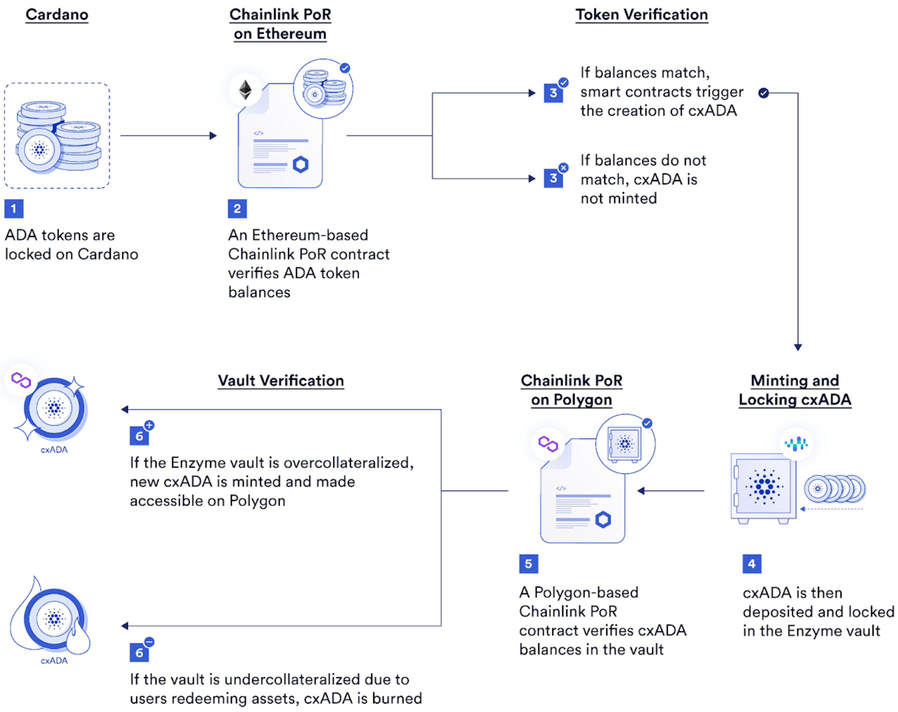
Proof of Reserve 🏦

> reliable and timely monitoring of reserve assets

- increases transparency
  - allows users to assess risk
  - trustless (everyone can check)
- backing of on-chain protocols/assets with off-chain reserves possible
- developers can ensure trust in their reserve management
  - huge oportunity for less known teams

Example: Ethereum Mainnet Reserves

Proof of Reserve 🏦



Cardano

1 ADA tokens are locked on Cardano

Chainlink PoR on Ethereum

2 An Ethereum-based Chainlink PoR contract verifies ADA token balances

Token Verification

3 If balances match, smart contracts trigger the creation of cxADA

3 If balances do not match, cxADA is not minted

Minting and Locking cxADA

4 cxADA is then deposited and locked in the Enzyme vault

Chainlink PoR on Polygon

5 A Polygon-based Chainlink PoR contract verifies cxADA balances in the vault

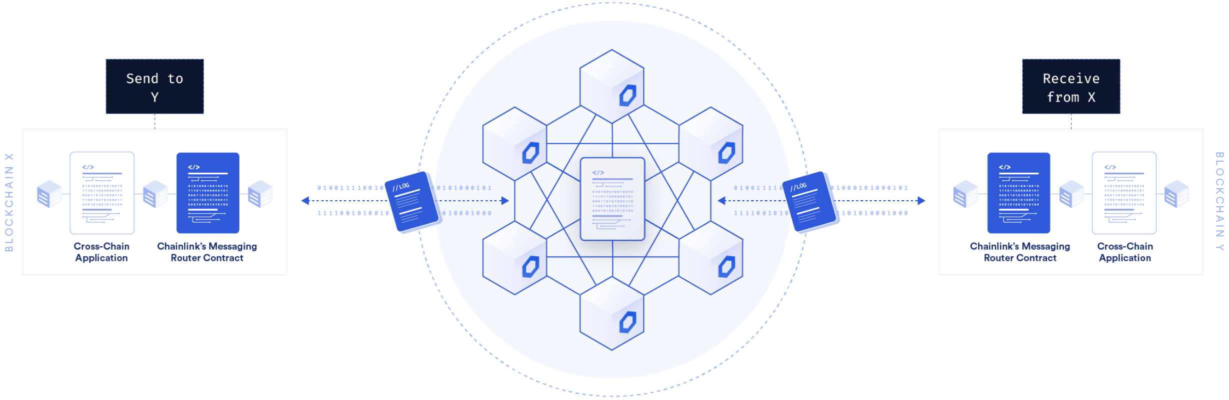
Vault Verification

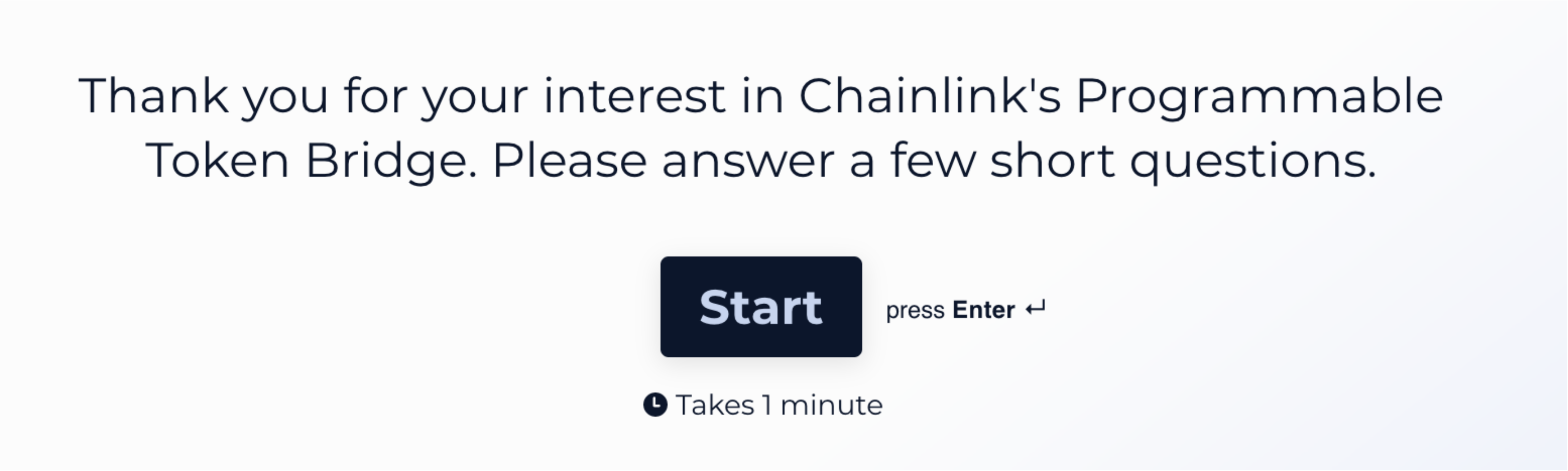
6 If the Enzyme vault is overcollateralized, new cxADA is minted and made accessible on Polygon

6 If the vault is undercollateralized due to users redeeming assets, cxADA is burned

Cross Chain Communication 💬

Send messages between Chain X and Y





Thank you for your interest in Chainlink's Programmable Token Bridge. Please answer a few short questions.

😢

Any API 📦

- enables smart contracts to access **ANY** external API
- uses decentralized oracle network

```
...

function requestVolumeData() public returns (bytes32 requestId) {
  Chainlink.Request memory req = buildChainlinkRequest(jobId, address(this), this.fulfill.selector);

  req.add('get', 'https://min-api.cryptocompare.com/data/pricemultifull?fsyms=ETH&tsyms=USD');

  req.add('path', 'RAW,ETH,USD,VOLUME24HOUR');

  // Multiply the result by 1000000000000000000 to remove decimals
  int256 timesAmount = 10**18;
  req.addInt('times', timesAmount);

  // Sends the request
  return sendChainlinkRequest(req, fee);
}

...
```

Functionality Recap 👨‍🏫

We can...

- query decentralized data feeds providing aggregated information
- generate verifiable & tamper proof random numbers
- automate smart contract executions off-chain
- expose & proof the status of a reserve
- enable cross chain data exchange & *even* transactions
- query any API on the internet

⇢ Chainlink provides developers with the necessary tools to enable "real-world usability" of smart contracts 💪

Arbol

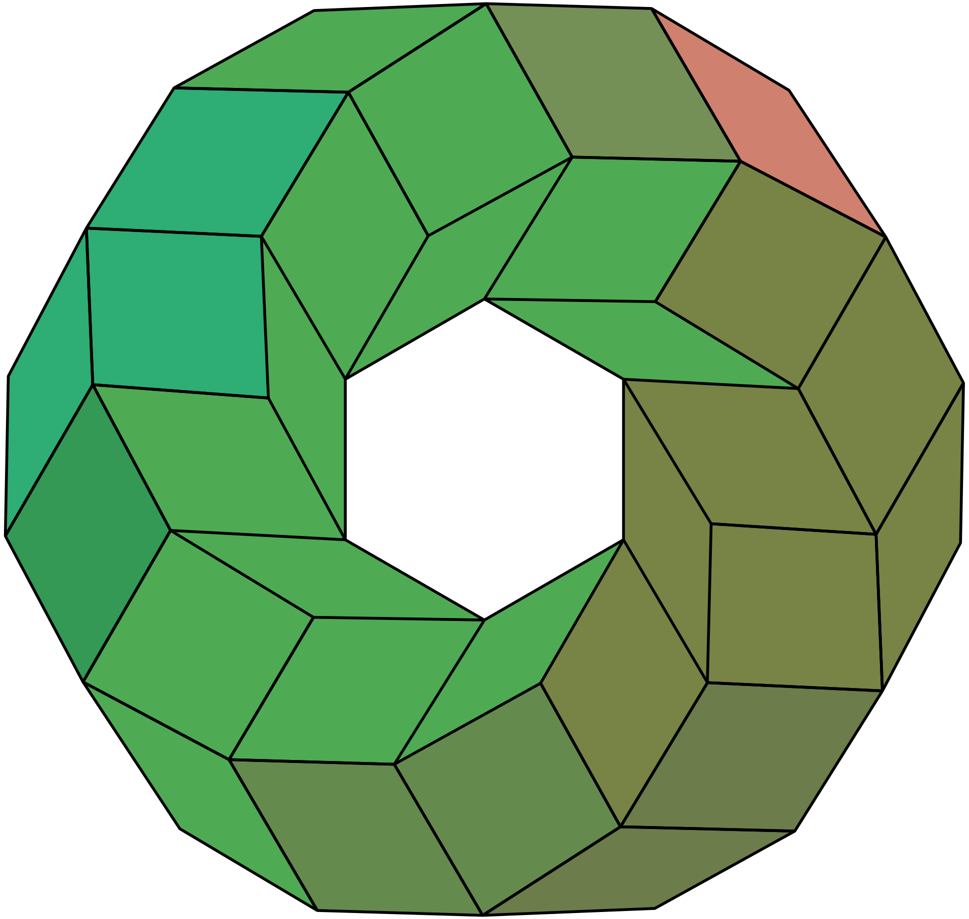
a short case study…

- farmers (…) insure their crop for loss
- once a loss threshold is met they get paid immediately

How is this special? 🧐

- Chainlink provides weather data to the contract
- no paperwork is needed, the smart contract pays automatically
- no more haggling with the insurance companies

expanded into Energy, Maritime & Hospitality



Thank you for your attention!

The presentation is available at `https://calwritescode.github.io/blockchain-oracles-2022/`