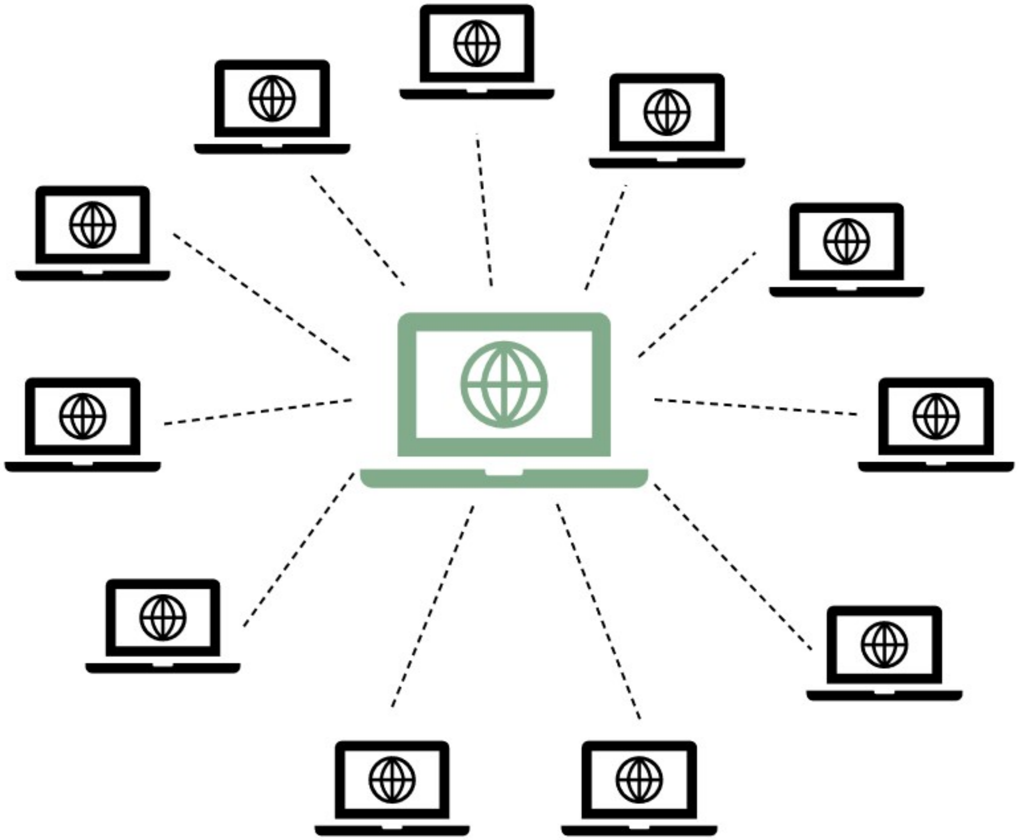
ORACLES IN ETHEREUMS VIRTUAL MACHINE

SEMINAR CRYPTOGRAPHY AND DATA SECURITY

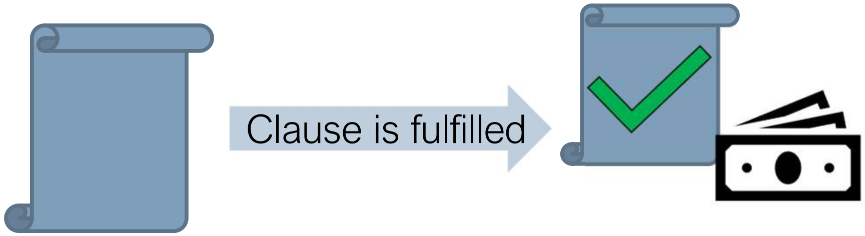
REMINDER EVM

- One single instance
- Each Node has a copy
- All Smart Contracts are on it
- Immutable



INTRODUCTION SMART CONTRACTS

- Contract written in Code (Solidity)
- Deployed to EVM -> Each has an Address
- User interacts by sending transactions
- Define rules which are automatically enforced
- Can’t be deleted or reversed
- https://remix.ethereum.org/
- Example: cupcake.sol

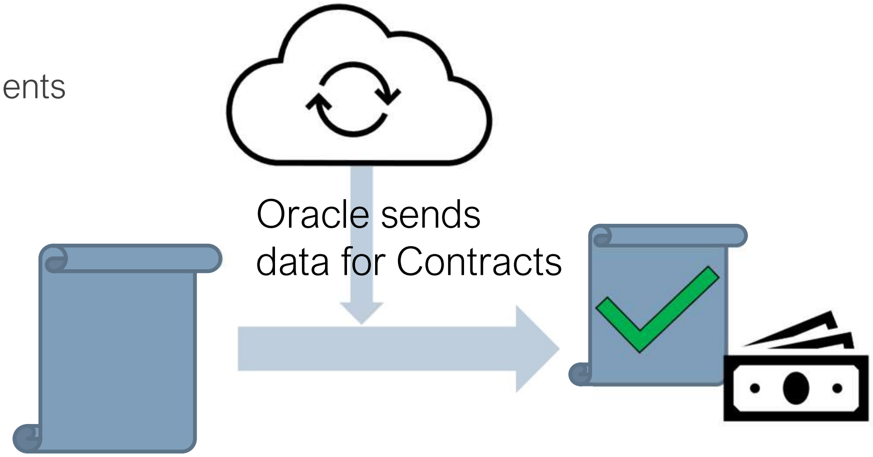


Contract with Clause

Automatic exchange of Goods and Services as defined

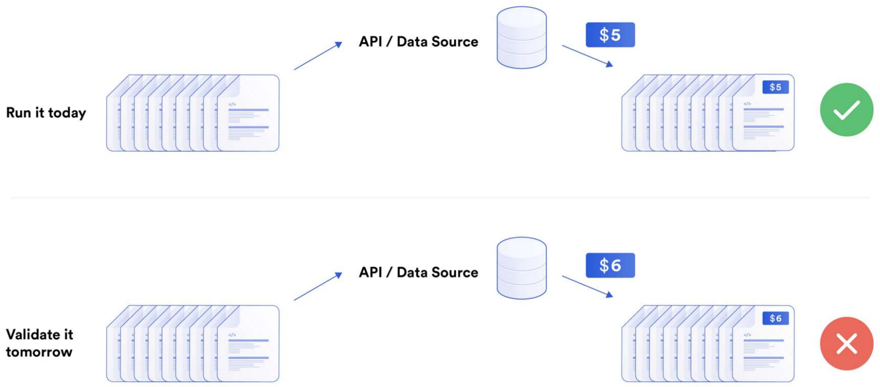
GOAL OF ORACLES: INTRODUCE REAL DATA INTO SMART CONTRACTS

- Contracts usually refer to real-world situations/elements
  - Exchange Rates (e.g., USD to CHF or ETH)
  - Production/Delivery Updates
  - Service Fulfillment
  - Weather
  - Event outcomes (i.e., Sport results)
  - …

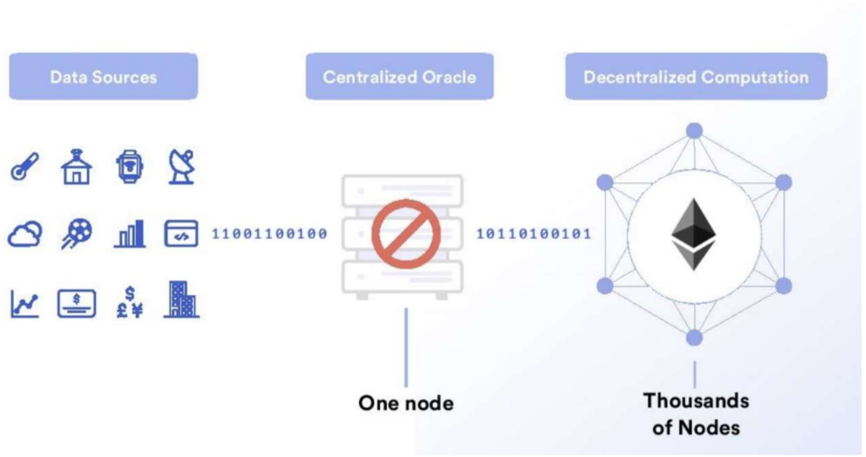


Oracle sends data for Contracts

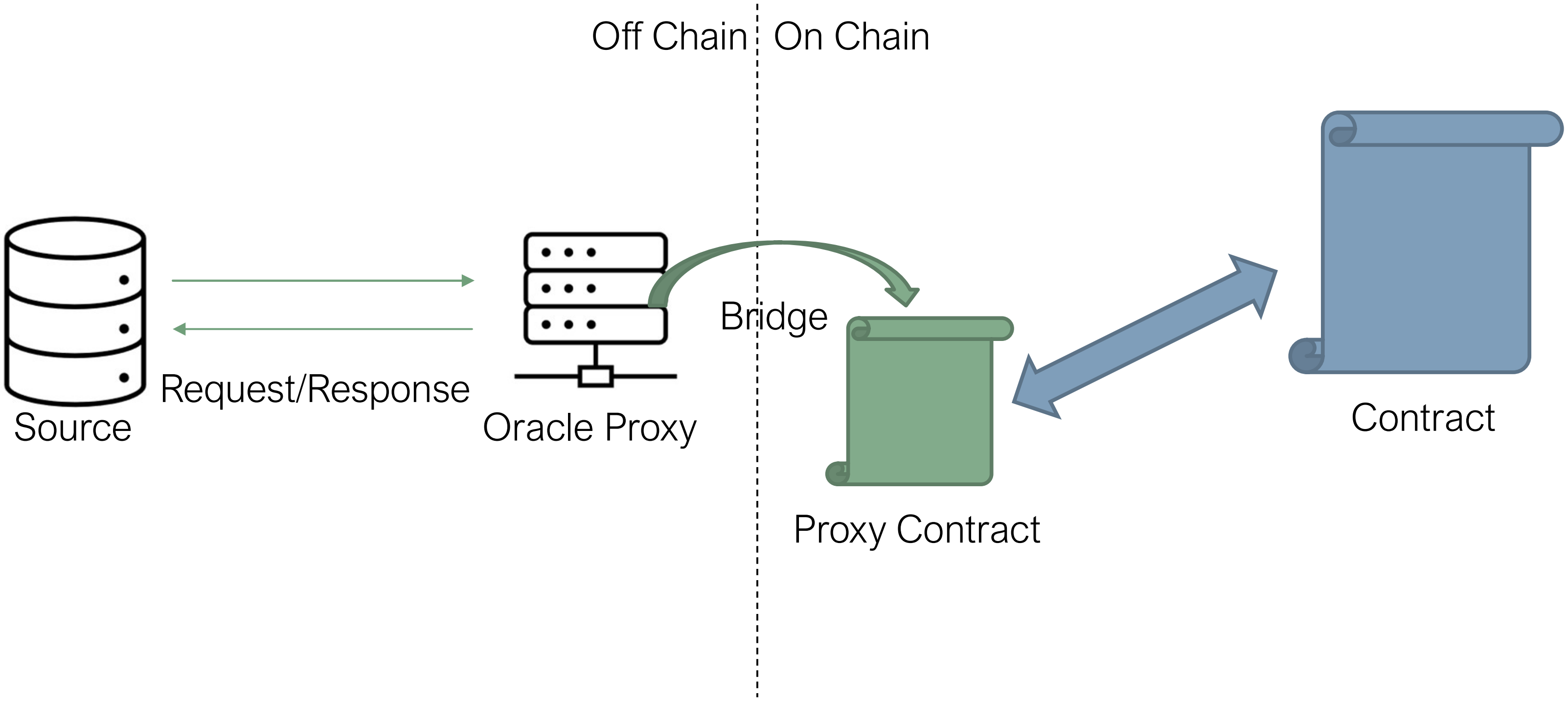
THE ORACLE PROBLEM



- Deterministic Blockchain
  - Each node must get the same result for the same input
  - Consensus can't be reached when the source is non-deterministic/changes its output depending on the time
- Using centralized Oracles nullifies the advantage of blockchain



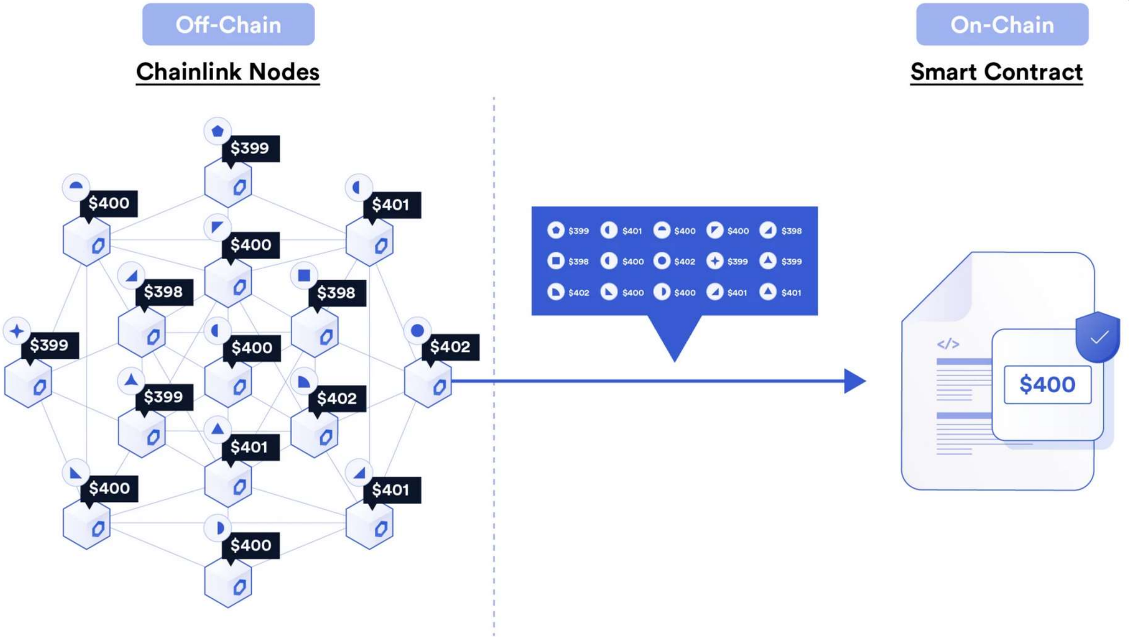
BASIC CONCEPT FOR ORACLES



SOLUTION TO THE ORACLE PROBLEM (CHAINLINK)

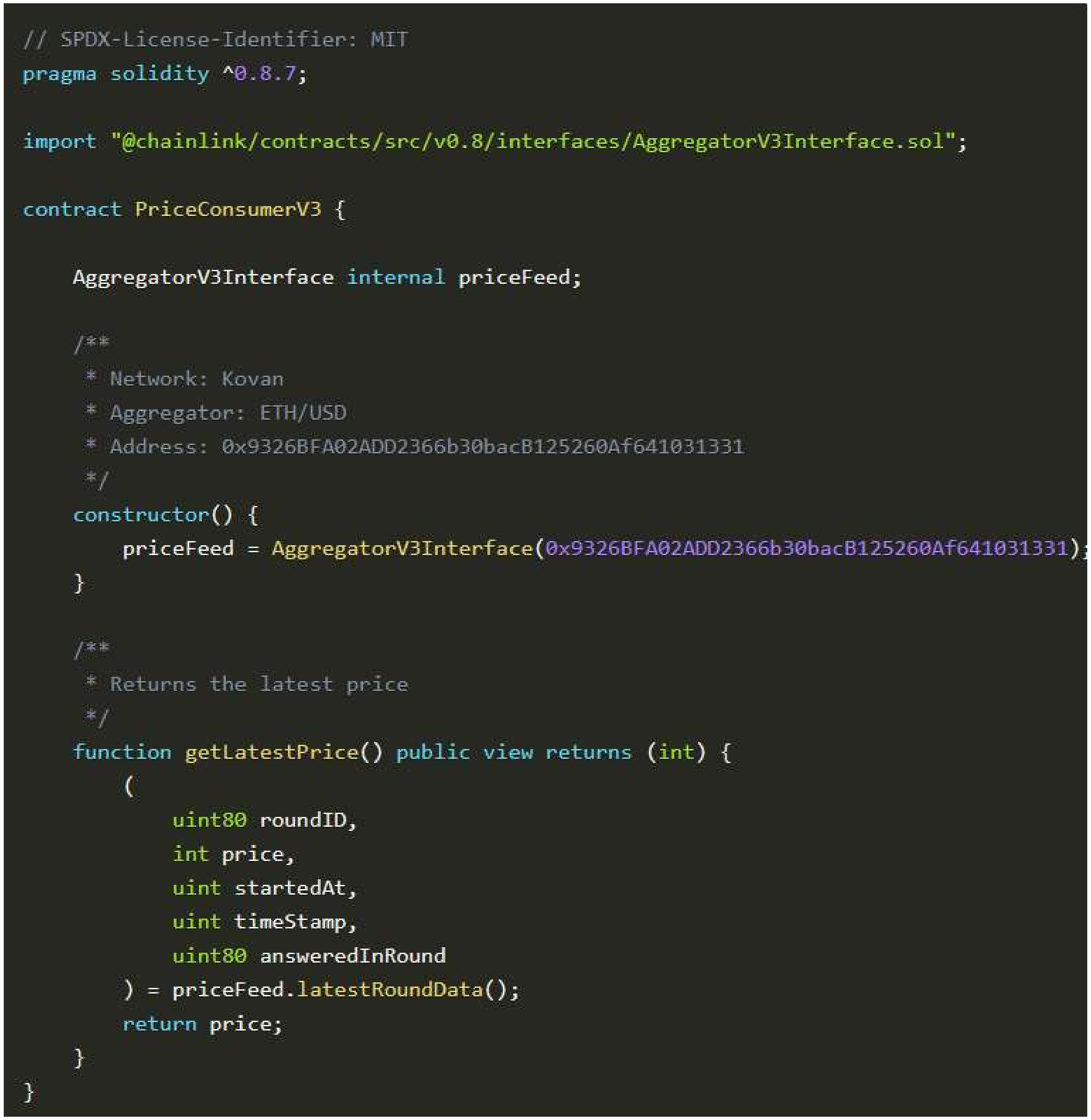
- Off-Chain Reporting
  - Chainlink Nodes off-chain (blockchain agnostic)
  - Elect a temporary leader
  - Leader requests signed data updates from each node
  - On request:
    - Leader sends result
    - If leader fails round robin kicks in until one node can send result to smart contract
- Incentive for Hosts: LINK Coins

Chainlink



EXAMPLE SMART CONTRACT WITH CHAINLINK ORACLE

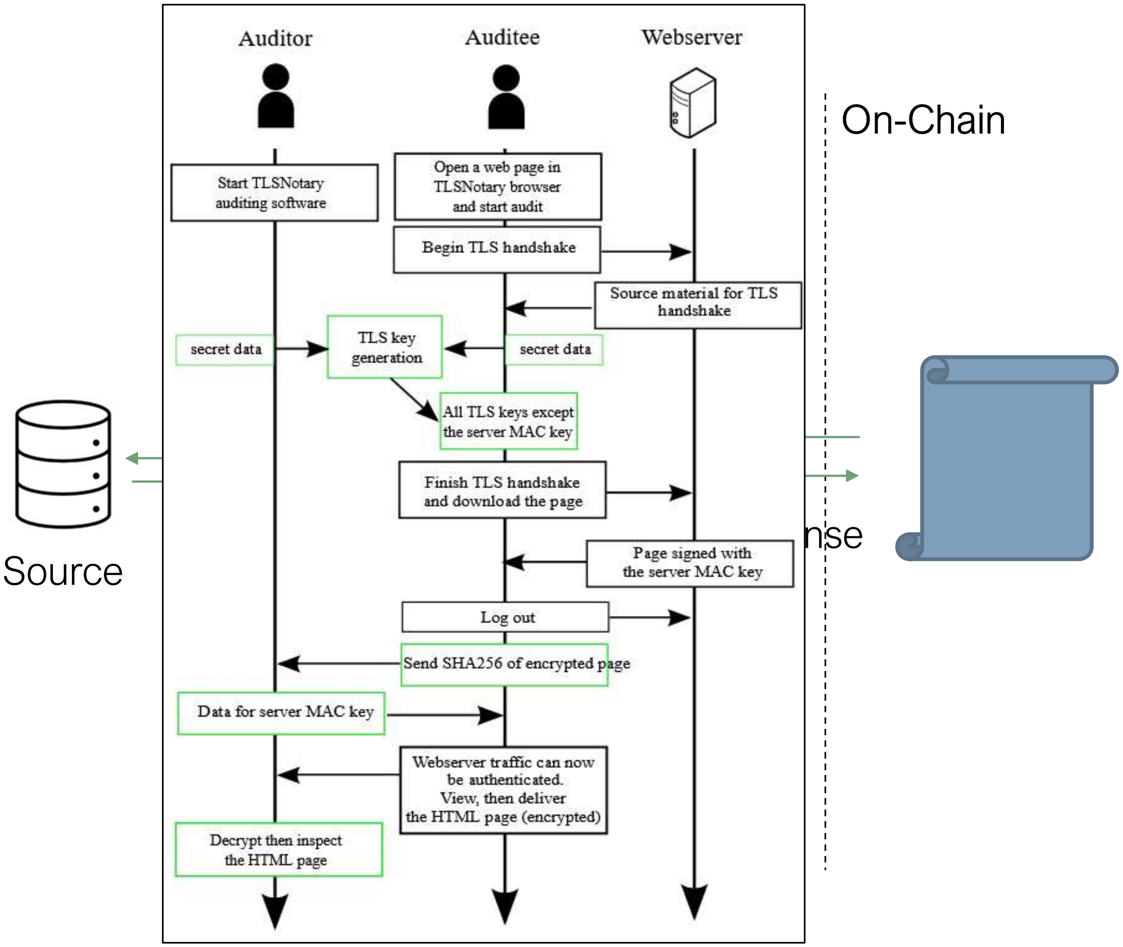
- https://remix.ethereum.org/#url=https://docs.chain.link/samples/PriceFeeds/PriceConsumerV3.sol



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

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

contract PriceConsumerV3 {

    AggregatorV3Interface internal priceFeed;

    /**
     * Network: Kovan
     * Aggregator: ETH/USD
     * Address: 0x9326BFA02ADD2366b30bacB125260Af641031331
     */
    constructor() {
        priceFeed = AggregatorV3Interface(0x9326BFA02ADD2366b30bacB125260Af641031331);
    }

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

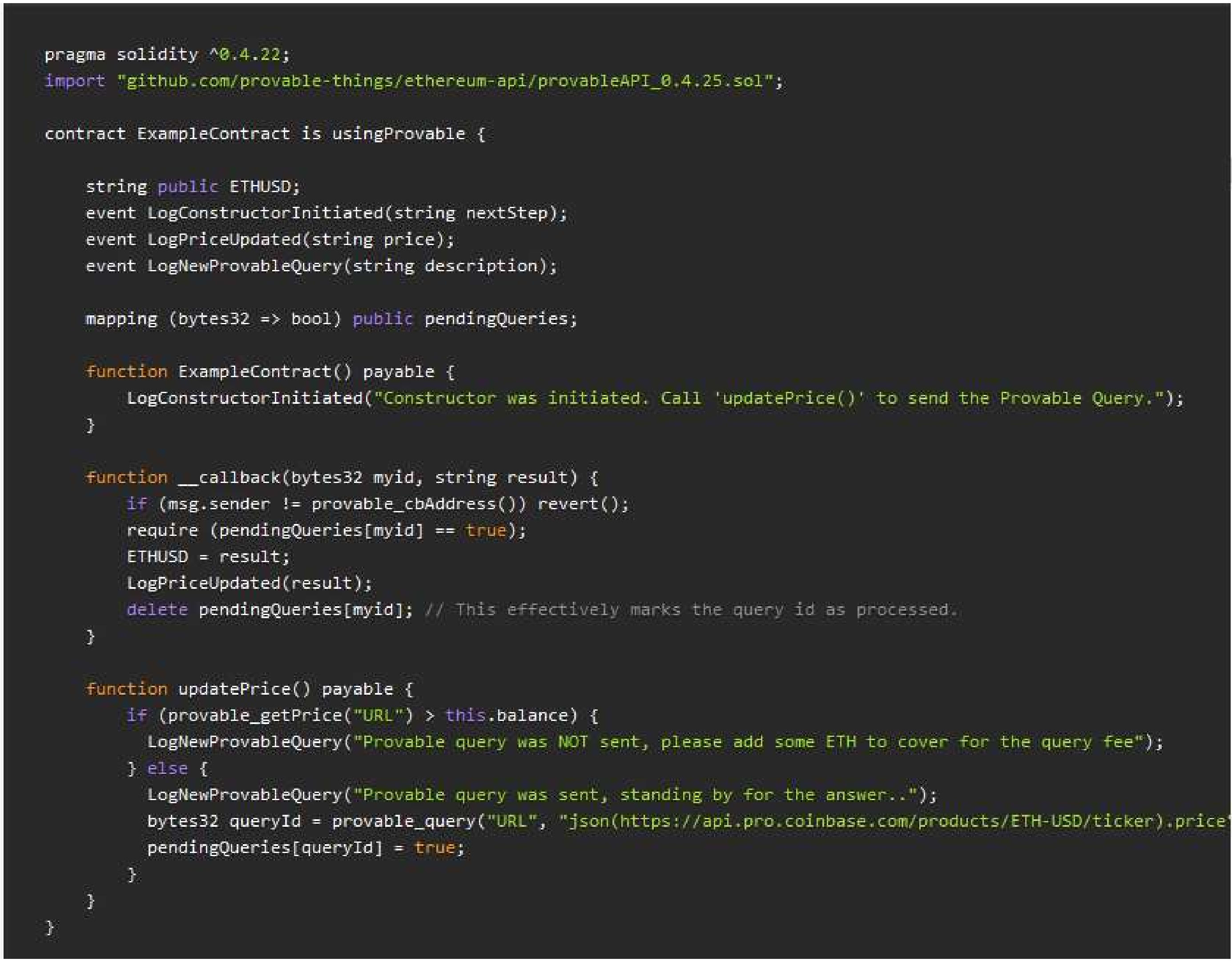
SOLUTION TO THE ORACLE PROBLEM (PROVABLE)

- Data is fetched by Provable from the original source
- Authenticity Proof shows that the data is correct and untampered


Auditor
Auditee
Webserver
Start TLSNotary auditing software
Open a web page in TLSNotary browser and start audit
Begin TLS handshake
Source material for TLS handshake
secret data
TLS key generation
secret data
All TLS keys except the server MAC key
Finish TLS handshake and download the page
Page signed with the server MAC key
Log out
Send SHA256 of encrypted page
Data for server MAC key
Webserver traffic can now be authenticated. View, then deliver the HTML page (encrypted)
Decrypt then inspect the HTML page
Source
On-Chain
nse

EXAMPLE SMART CONTRACT WITH PROVABLE ORACLE

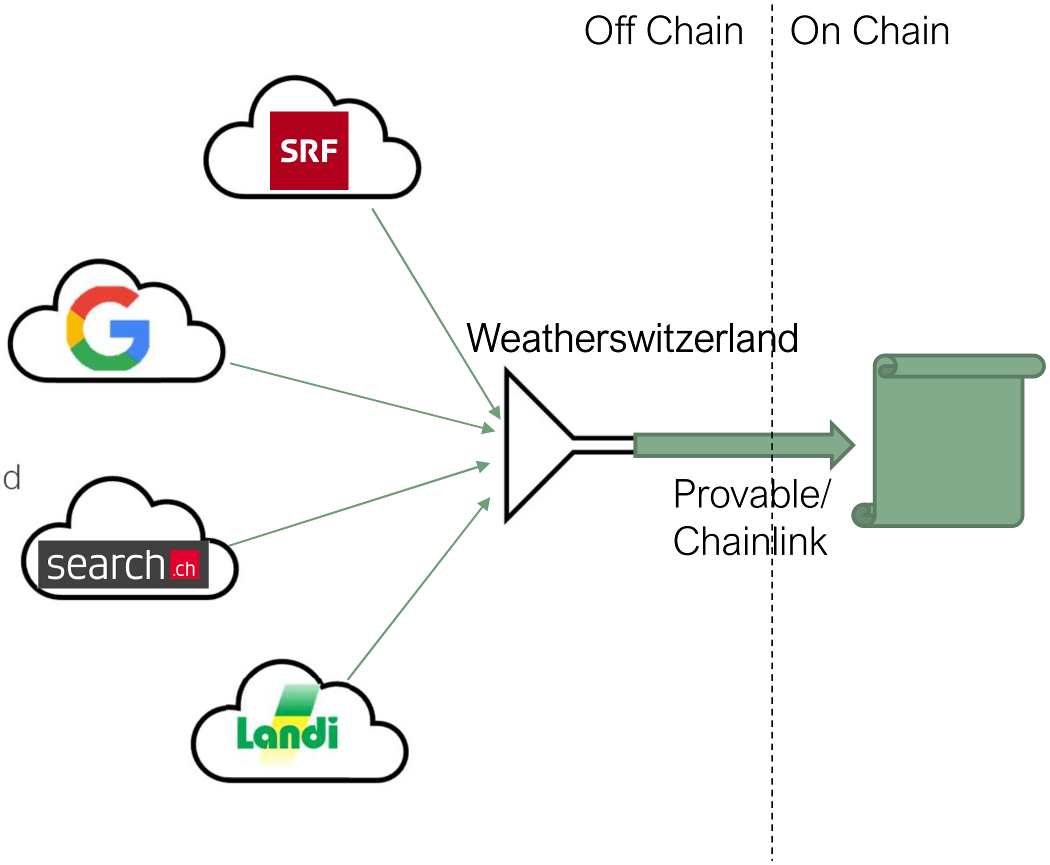
- https://docs.provable.xyz/#ethereum-quick-start



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

contract ExampleContract is usingProvable {

    string public ETHUSD;
    event LogConstructorInitiated(string nextStep);
    event LogPriceUpdated(string price);
    event LogNewProvableQuery(string description);

    mapping (bytes32 => bool) public pendingQueries;

    function ExampleContract() payable {
        LogConstructorInitiated("Constructor was initiated. Call 'updatePrice()' to send the Provable Query.");
    }

    function __callback(bytes32 myid, string result) {
        if (msg.sender != provable_cbAddress()) revert();
        require (pendingQueries[myid] == true);
        ETHUSD = result;
        LogPriceUpdated(result);
        delete pendingQueries[myid]; // This effectively marks the query id as processed.
    }

    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..");
          bytes32 queryId = provable_query("URL", "json(https://api.pro.coinbase.com/products/ETH-USD/ticker).price
          pendingQueries[queryId] = true;
        }
    }
}
```

MY PROJECT

- Using Oracle to get current Temperature in Switzerland
- Simplified example on how Chainlink operates
- https://weatherswitzerland.herokuapp.com/


Off Chain
On Chain
SRF
search.ch
Landi
Weatherswitzerland
Provable/
Chainlink