Chainlink

Smart Contract Developer Bootcamp

Day 1

Chainlink

Smart Contract Developer Bootcamp

Day 1

Introduction to Blockchain, Ethereum, Smart Contracts, and Chainlink

Harry Papacharissiou

Developer Advocate, Chainlink Labs

harry@chainlinklabs.com

@Pappas9999

harrypapacharissiou

Chainlink

Smart Contract Developer Bootcamp Outline

Day 1: Introduction

Learn about blockchain, smart contracts and solidity, followed by examples of how to use Chainlink Price Feeds, Any-API and VRF in your smart contracts. Working with Solidity

Day 2: Development Environments

Learn how to use the Hardhat or Brownie development environment, followed by a guide on using our starter kits to create smart contracts that use Chainlink. Refining your project with local deployments and testing

Chainlink

Housekeeping Rules

- 3-4 hour session, broken up into sections, with breaks
- Presentation, followed by questions, demo and exercise
- Questions can be asked at the end of each section
- Zoom: If you need help during a practical exercise, raise your hand and an instructor will eventually get to you and send you an invite to a private breakout session ✋
- YouTube: Technical questions can be asked in the chat or in the #developer-bootcamp channel in the Chainlink Discord

Day 1 Exercises

- Exercise requirements:
  - MetaMask
  - Funding your account with ETH and LINK
- Faucet: https://linkfaucet.protofire.io/kovan

Agenda: Day 1

1. **Blockchain**
2. Smart Contracts
3. Ethereum and Solidity
4. The Oracle Problem and Chainlink
5. Chainlink Data Feeds
6. Accessing any API using Chainlink
7. Obtaining verifiably random numbers using Chainlink VRF

> *A beginning is the time for taking the most delicate care that the balances are correct*

- Frank Herbert
Dune

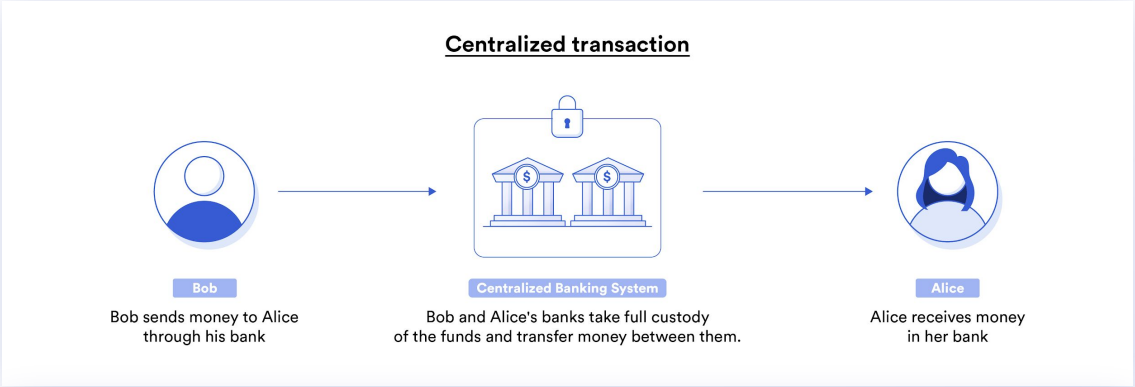
Chainlink

Blockchain

What is a blockchain?

A blockchain is highly **secure**, **reliable**, and **decentralized network** that stores data, exchange values, and record transaction activity in a **shared ledger** that is **not controlled by any central authority**, but instead maintained by computers all around the world.

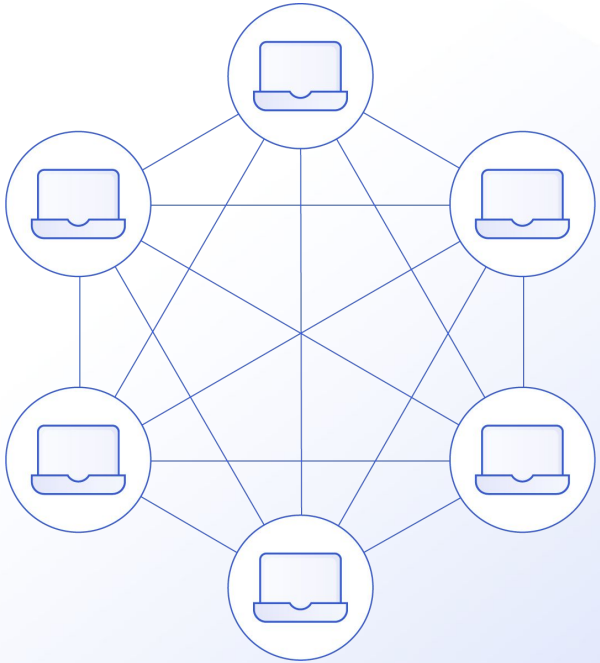
What is a blockchain?



Chainlink

Decentralization

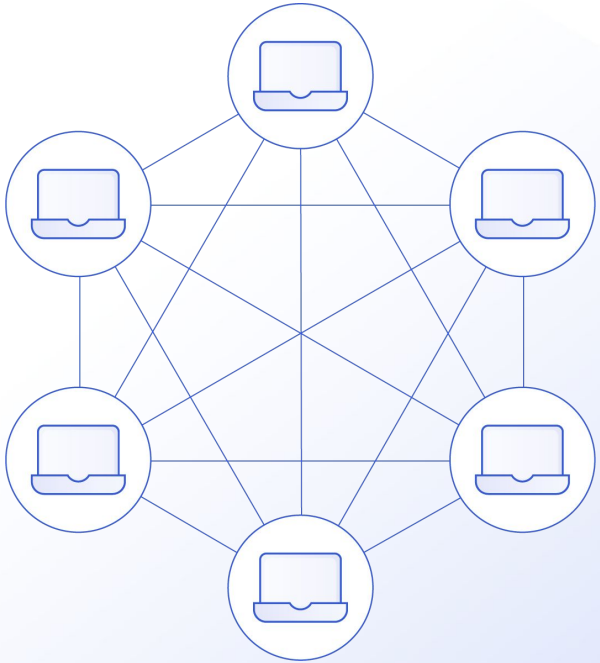
- Architectural
- Political
- Logical



Chainlink

Benefits of Decentralization

- Fault tolerance
- Attack resistance
- Collusion resistance



Blockchain Demo

Smart Contracts

What is a Contract and What is its Purpose?

Contract:

A binding agreement between two or more persons or parties.

Merriam-Webster

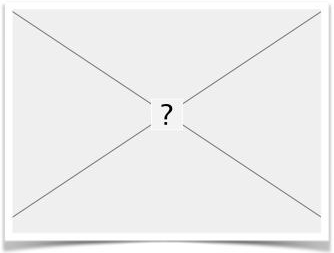
Contract:

A legal document that states and explains a formal agreement between two different people or groups.

Cambridge Dictionary

Chainlink

The Next Stage: Technologically Enforced Contracts



Telegraph Agreements (Electronically Signed) 1869

Telex Machines (Telecom Based) 1930s

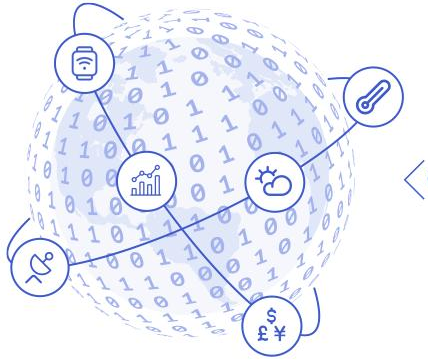
Digital Agreements (Internet Based) 1980s to Present

Smart Contracts (Blockchain Based) 2009 to Present

Chainlink

Digital Agreements

Performance Data



0010111011

Contract Terms

Participants

1100100101

0110010010

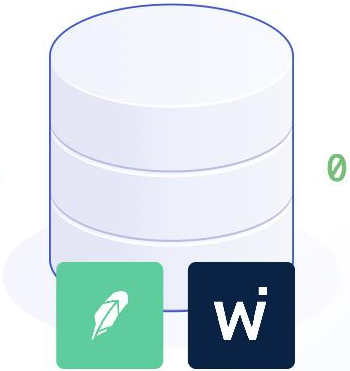
Chainlink

CeFi Financial Products Work Well in Predictable Conditions

Average Users

01011011

Centralized Finance



01011011

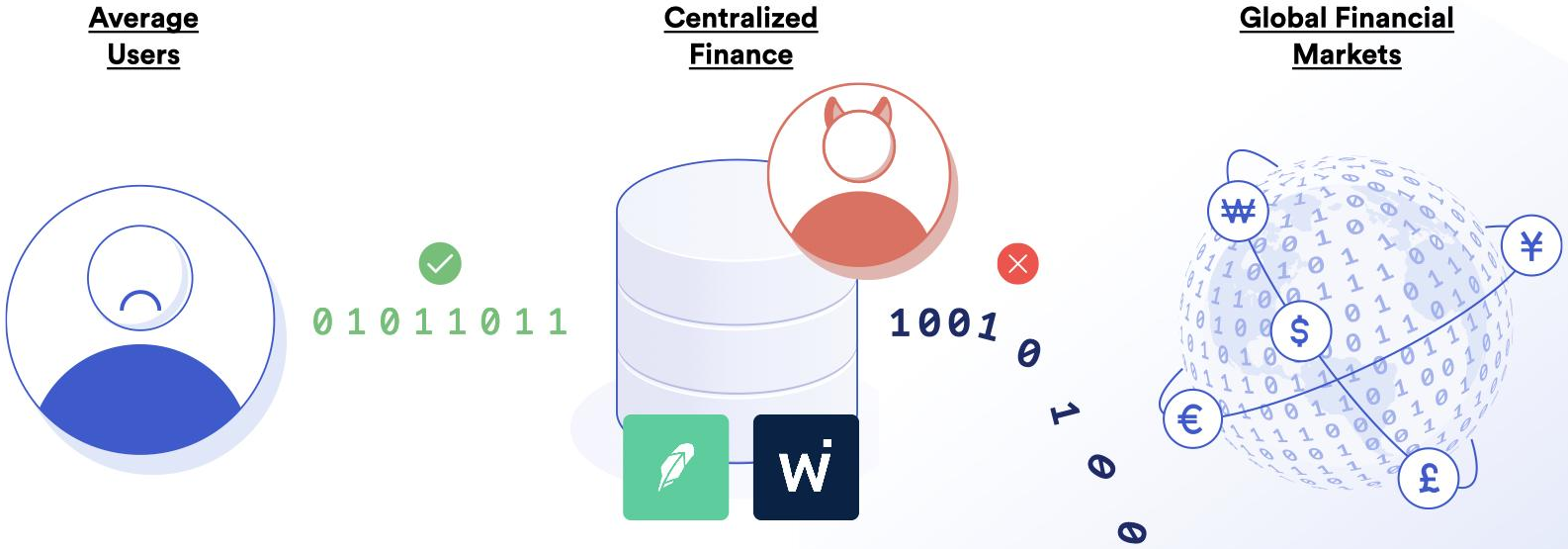
Global Financial Markets

MarketWatch

Robinhood day traders are squeezing the hedge funds — here's why it may continue

Chainlink

CeFi Financial Products Will Not Support the Average User



VICE

Robinhood Stops Users From Trading GameStop Stocks, Other Reddit YOLO Picks

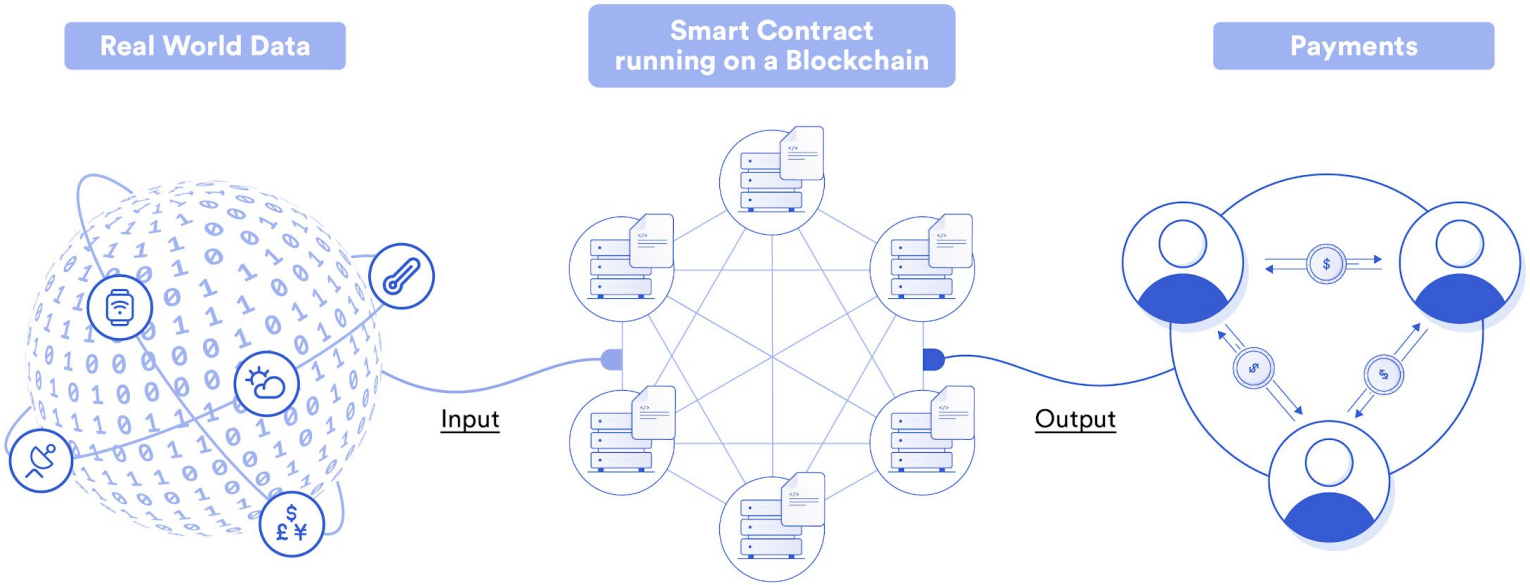
Chainlink

Smart Contracts

A Smart Contract is **a self-executing contract** with the terms of the agreement being **directly written into computer code**

Programmatically implement **a series of if-then rules** without the need for third-party human interaction

Smart Contracts Run on the blockchain



Smart Contracts are Superior Digital Agreements

Security

Tamper-proof digital agreements that can't be influenced by a single party

Chainlink

Smart Contracts are Superior Digital Agreements

Security

Tamper-proof digital agreements that can't be influenced by a single party

Guaranteed Execution

Execution and enforcement of the agreement is performed by an always-on decentralized network

Chainlink

Smart Contracts are Superior Digital Agreements

Security

Tamper-proof digital agreements that can't be influenced by a single party

Guaranteed Execution

Execution and enforcement of the agreement is performed by an always-on decentralized network

Transparency

Transparency of the agreement and it's enforcement is unavoidably built-in

Chainlink

Smart Contracts are Superior Digital Agreements

1

Security

Tamper-proof digital agreements that can't be influenced by a single party

2

Guaranteed Execution

Execution and enforcement of the agreement is performed by an always-on decentralized network

3

Transparency

Transparency of the agreement and it's enforcement is unavoidably built-in

4

Trust Minimization

Reduced counterparty risk due to neither party having control of the agreements execution or enforcement

Chainlink

Smart Contracts are Superior Digital Agreements

1

Security

Tamper-proof digital agreements that can't be influenced by a single party

2

Guaranteed Execution

Execution and enforcement of the agreement is performed by an always-on decentralized network

3

Transparency

Transparency of the agreement and it's enforcement is unavoidably built-in

4

Trust Minimization

Reduced counterparty risk due to neither party having control of the agreements execution or enforcement

5

Efficiency

All these traits provide an opportunity to migrate existing manual processes to be automated

Chainlink

Smart Contracts Will Solve Society's Critical Trust Issues

> **Counterparty Risk:** the likelihood or probability that one of those involved in a transaction might default on its contractual obligation.

Paper Guarantees (Brand Based)

- **Counterparty risk is high and opaque**
- Transparency is purposefully removed
- Interest yields are low and going lower

Cryptographic Guarantees (Math Based)

- **Counterparty risk is low and transparent**
- Transparency is unavoidably built-in
- Interest yields are consistently high

https://hackernoon.com/what-is-the-math-behind-elliptic-curve-cryptography-f61b25253da3

Chainlink

Ethereum and Solidity

Ethereum

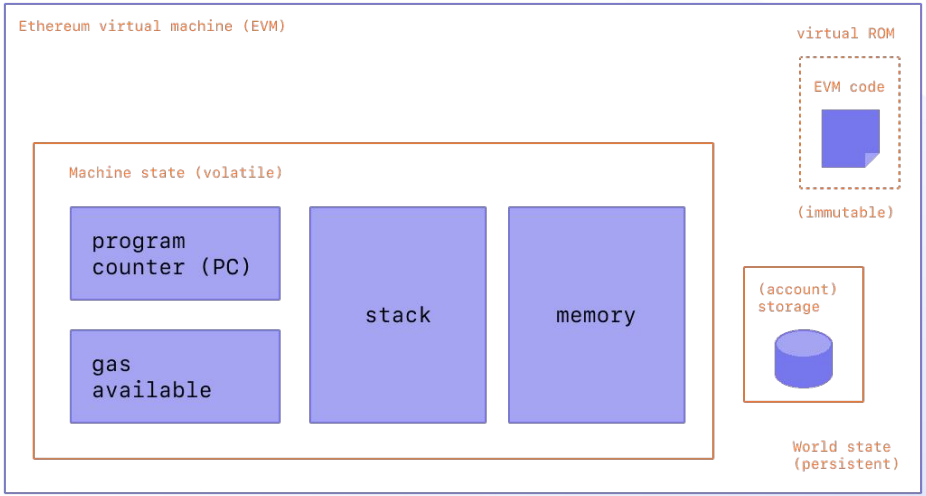
- Permissionless (public) blockchain
  - Open source, blockchain platform with smart contracts
  - Has a native cryptocurrency called ‘Ether’ or ‘ETH’
- Transaction-based state machine
- Currently uses Proof of Work consensus mechanism, but in the process of transitioning to Proof of Stake

Ethereum

- Nodes
- Accounts
- Transactions
- Pending Transactions (mempool)

The Ethereum Virtual Machine

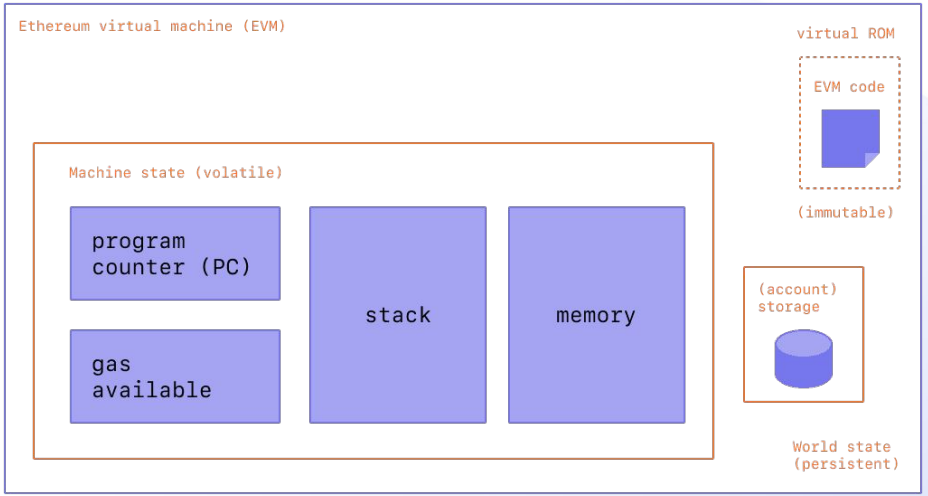
- Turing-complete 256 bit Virtual machine
- Built into the Ethereum protocol
- Determines the outcome of executed contract logic
- Stack-based VM that uses bytecode to encode instructions



Chainlink

The Ethereum Virtual Machine

- Ethereum smart contracts are executed in the EVM
  - Contract logic is programmed via a language such as Solidity
  - Contracts are then compiled to EVM bytecode



Chainlink

Ether

- Native cryptocurrency of the Ethereum network
- Symbol is ETH
- Used for transaction fees and computation (executing smart contract code)

Tokens

- Representation of any tradable good or service, or a unit of account
- Sample token use cases:
  - Virtual currencies
  - Utility token
- The ER-C20 token standard allows building tokens
- The Chainlink token is an ERC-677 token

Solidity

- Contract-oriented high-level programming language
  - Syntax is similar to JavaScript and C#
  - Contracts and inheritance similar to OOP
  - Libraries
- Official documentation

Solidity Code Compilation

- Contract is compiled by the Solidity compiler
  - Bytecode (EVM instructions)
  - ABI definition
- Application Binary Interface (ABI) describes the contract's interface
  - Public methods
  - Inputs
  - Return values
  - Events
- EVM is where the contracts are executed

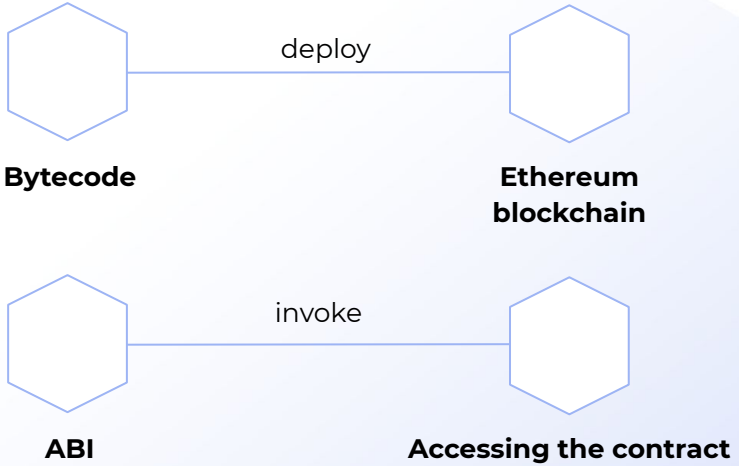
ABI Output:

```
[
    {
        "inputs": [],
        "name": "retrieve",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "num",
                "type": "uint256"
            }
        ],
        "name": "store",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    }
]
```

Chainlink

Solidity Code Compilation

- Contract is published by uploading the bytecode to the blockchain
- To access the contract, you need the ABI definition to know what inputs and outputs the contract works with



Chainlink

Chainlink

Gas Payment

- Executing smart contract functions isn't always free, execution sometimes costs gas
- Gas is the internal unit for keeping track of execution cost
- Gas Cost measures the consumed computational resources
  - Each CPU instruction in EVM costs gas:
  - Each persistent storage write also costs gas
- Eg: Writing 32 bytes into a contract memory costs 80000 gas
- Gas price is how much someone is willing to pay per unit of Gas Cost
  - Can be negotiated, but must be > 0
  - Eg: 1 gas = 20 Gwei

Chainlink

Gas Payments

- Gas is measured and paid for at each contract execution
- Reading data from the blockchain is free
- The gas price is set by the transaction caller
  - The lower the price, the more time it could take for a miner to include the transaction in a block
- Can use gas calculators, eg Eth Gas Station

Chainlink

Gas Limit

- Maximum allowed gas to be used in a transaction
- Limits must be set to avoid never ending function execution
- If the limit is reached, contract execution will be terminated
  - Any changes to the contract state will revert
  - User will lose their ETH spent on gas

Chainlink

Ethereum Block Limits

- Blocks are mined roughly every 15 seconds
- Each block has a 15M gas limit
- Usually equates to 150-200 transactions per block

Operations and Gas

Operations that cost gas:

- Anything that changes blockchain state
  - Writing to contract memory
  - Sending Ether
  - Execution of contract logic
  - Writing data to the blockchain

Free operations:

- Anything not requiring changes to the blockchain
  - Reading blockchain data (contracts, accounts etc)

Gas Auctions: Bus Analogy

Gas Auctions: Bus Analogy

Contracts and Functions

- Contracts in Solidity are like classes in OOP programming
- Contracts can hold:
  - Data and functions
- Contract execution is paid with gas

```
contract Storage {

    uint256 number;

    /**
     * @dev Store value in variable
     * @param num value to store
     */
    function store(uint256 num) public {
        number = num;
    }

    /**
     * @dev Return value
     * @return value of 'number'
     */
    function retrieve() public view returns (uint256){
        return number;
    }
}
```

Chainlink

Chainlink

Function Visibility

- Public:
  - Can be called by anyone
  - For public variables, an automatic getter function is generated
- External:
  - Designed to be called externally by other contracts and systems
  - Cannot be called internally
- Internal
  - Accessed internally only
- Private
  - Accessible in the contract only

Function Parameters and Return Values

- Function with a uint parameter and no return value:

```
function store(uint256 num) public {
    number = num;
}
```

- Function with no parameters returning a value:

```
function retrieve() public view returns (uint256){
    return number;
}
```

Data Types in Solidity

- Similar to other programming languages:
  - Integers and unsigned integers
    - No floating-point/decimal numbers
    - Can do math operations
  - Boolean (true/false)
  - String: UTF8 encoded text, “Chainlink”
  - Address: Ethereum address (40 character hex, encoded with checksum)
    - 0xFCadTa6da569DF655M70DEd06cb7A1b2Ccd1D3AG
  - Bytes/Bytes8...Bytes32 - storing string data as bytes
  - Enumerated Type
    - Enables user defined types

```
enum Direction {Up, Down, Left, Right}
Direction playerDirection = Direction.Left;
```

Chainlink

Data Types in Solidity: Arrays

- Compile-time fixed size, or dynamic
- Can hold arbitrary types
- Members
  - Length: return the number of elements
  - Push: append new elements to the array

```
//fixed size
uint8[3] memory numbers = [1, 2, 3];
numbers[0] = 100;
```

```
//Dynamic array (variable size)
uint24[] numbers;
function addNum(uint24 _num) public returns (uint)
{
    numbers.push(_num);
  return numbers.length;
}
```

Data Types in Solidity: Mappings

- Key-value pair
  - Key type: Any type except for a mapping, dynamically sized array, contract, struct or enumeration
  - Value type: Anything
- The key isn't stored, its kekkak256 (hash) is used to look up the value

```
mapping (string => address) public accounts;
accounts["Harry"] =
  0xF7b4ef69E7Cf13C205566345CcFAd1aB5fdCc49F;
```

Chainlink

Data Types in Solidity: Structs

- Collection of primitive types
  - Has some restrictions (eg can't contain a member of its own type)
- Can store structs inside mappings and arrays
- Direct access to members of a struct

```
struct Account {
  address addr;
  uint amount;
}

Account acc = Account({
  addr: 0xF7b4ef69E7Cf13C205566345CcFAd1aB5fdCc49F,
  amount: 20});

acc.amount += 20;
```

Exercise 1: My First Smart Contract

- Remix: http://remix.ethereum.org/
- 30 minutes

Demo : Arrays and Mappings in Solidity

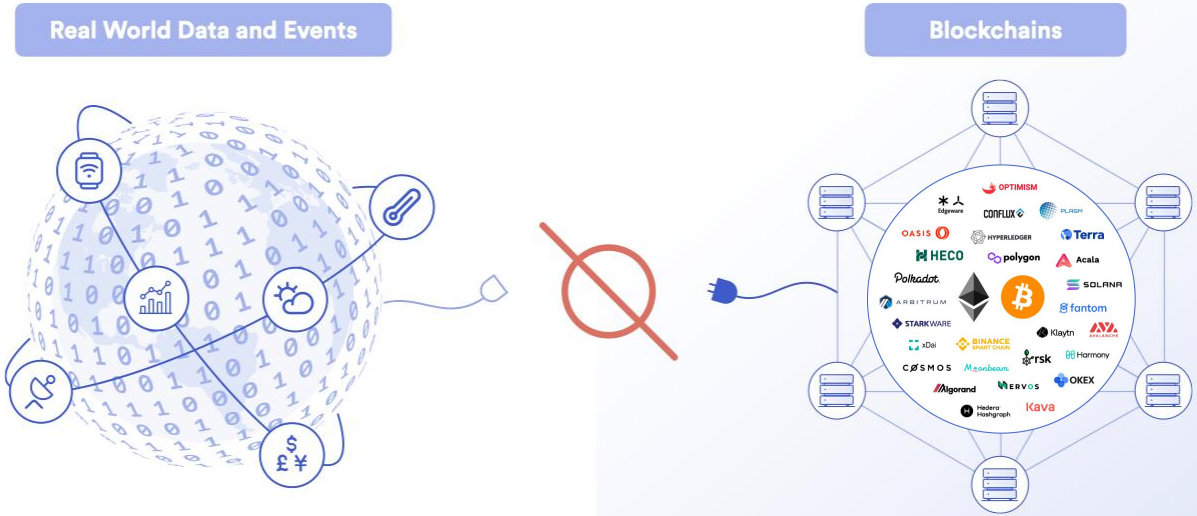
Exercise 2: Arrays and Mappings in Solidity

- Remix: http://remix.ethereum.org/
- 30 minutes

The Oracle Problem and Chainlink

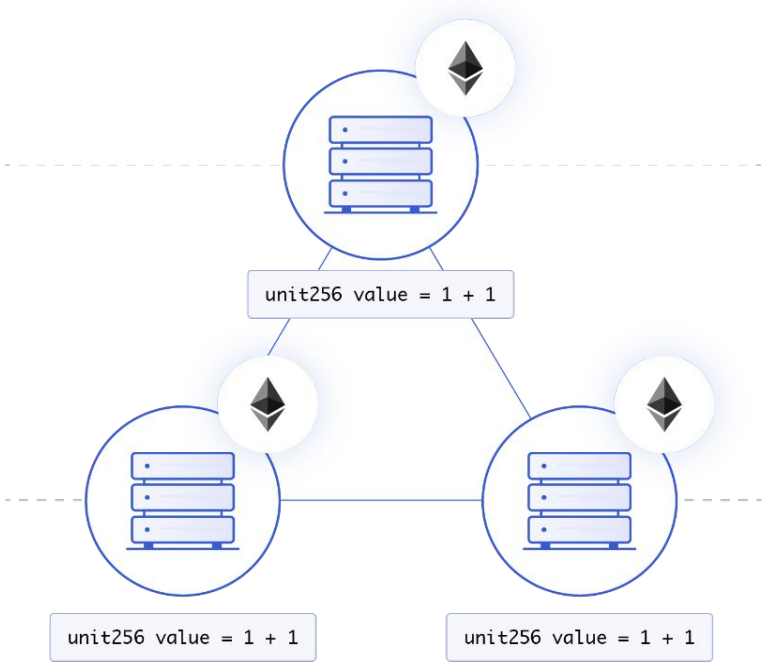
The “Oracle Problem” for Smart Contracts

Smart Contracts are unable to connect with external systems, data feeds, APIs, existing payment systems or any other off-chain resources on their own.



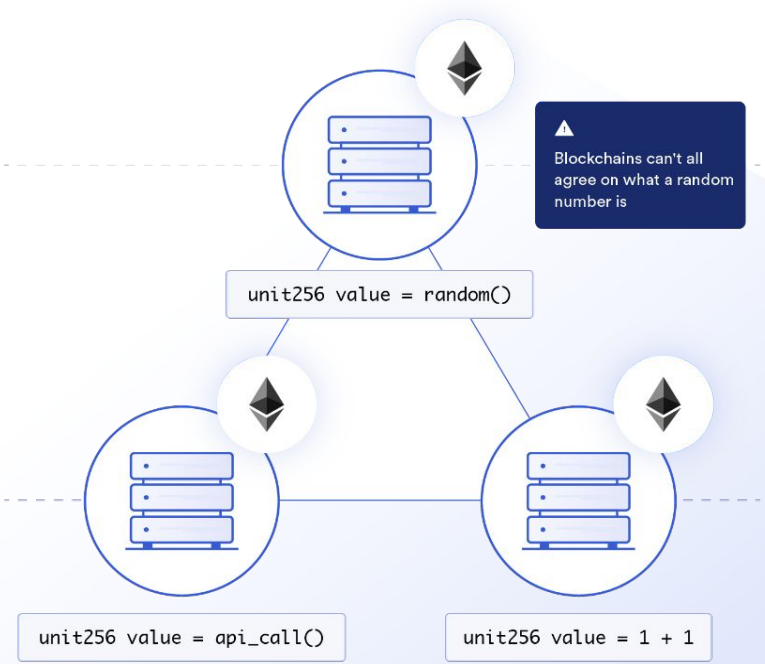
Chainlink

Deterministic



Consensus reached

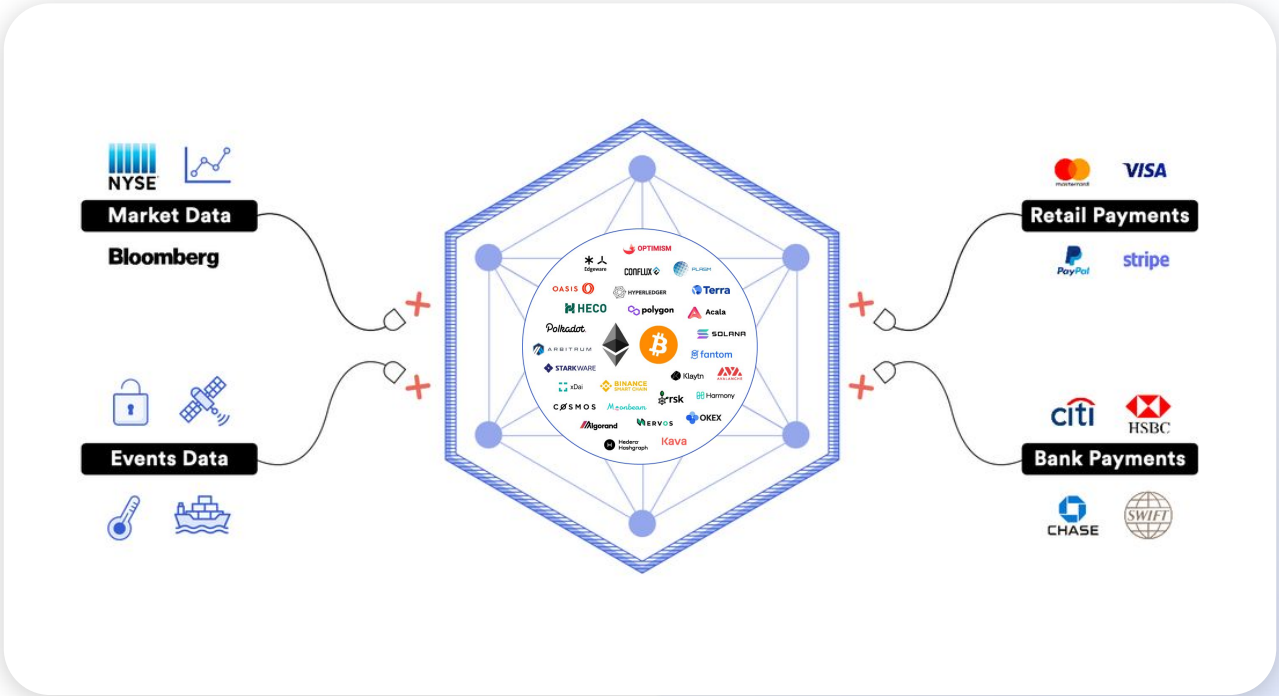
Non-deterministic



Blockchains can't all agree on API calls

Consensus not reached

The Smart Contract Connectivity Problem



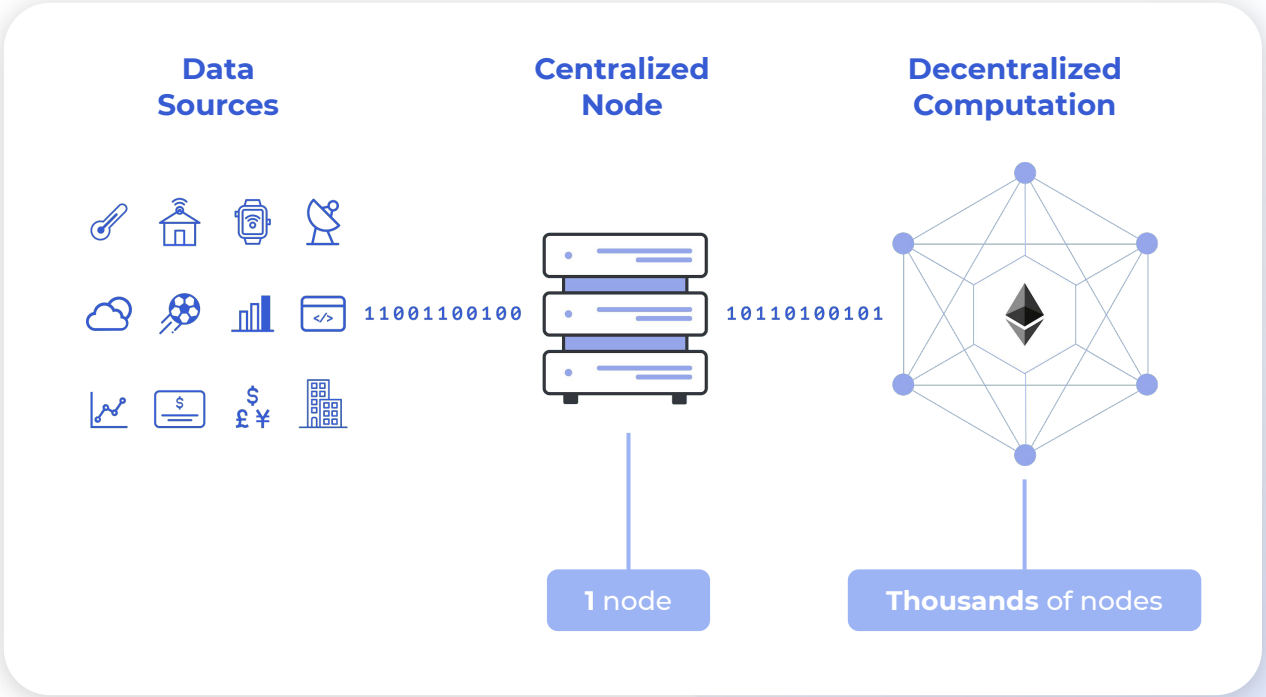
Chainlink

The Smart Contract Connectivity Problem

Blockchain Oracle:

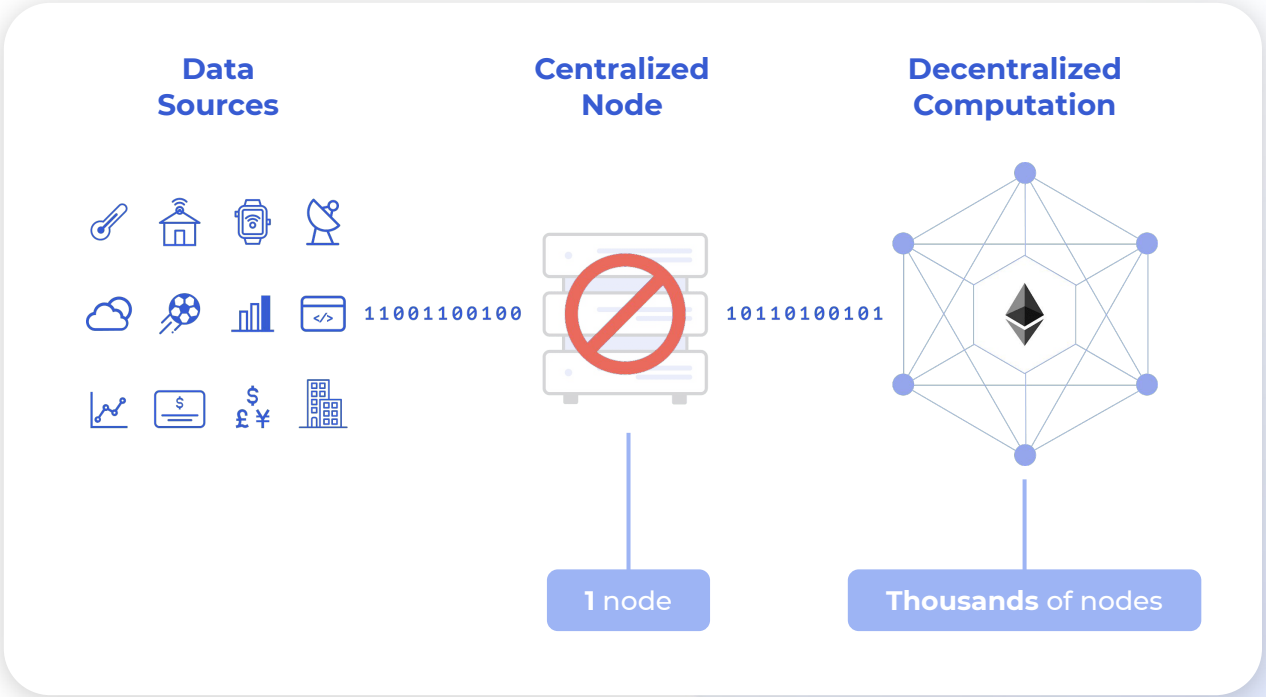
Any device that interacts with the off-chain world to provide external data or computation to smart contracts.

Centralized Oracles are a Point of Failure



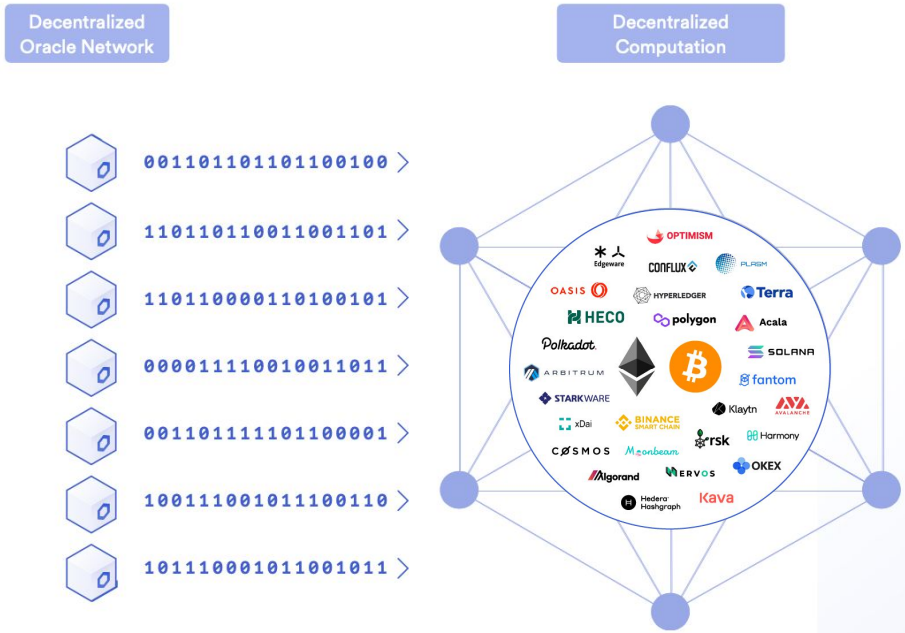
Chainlink

Centralized Oracles are a Point of Failure



Chainlink

A Decentralized Oracle Network



Decentralization

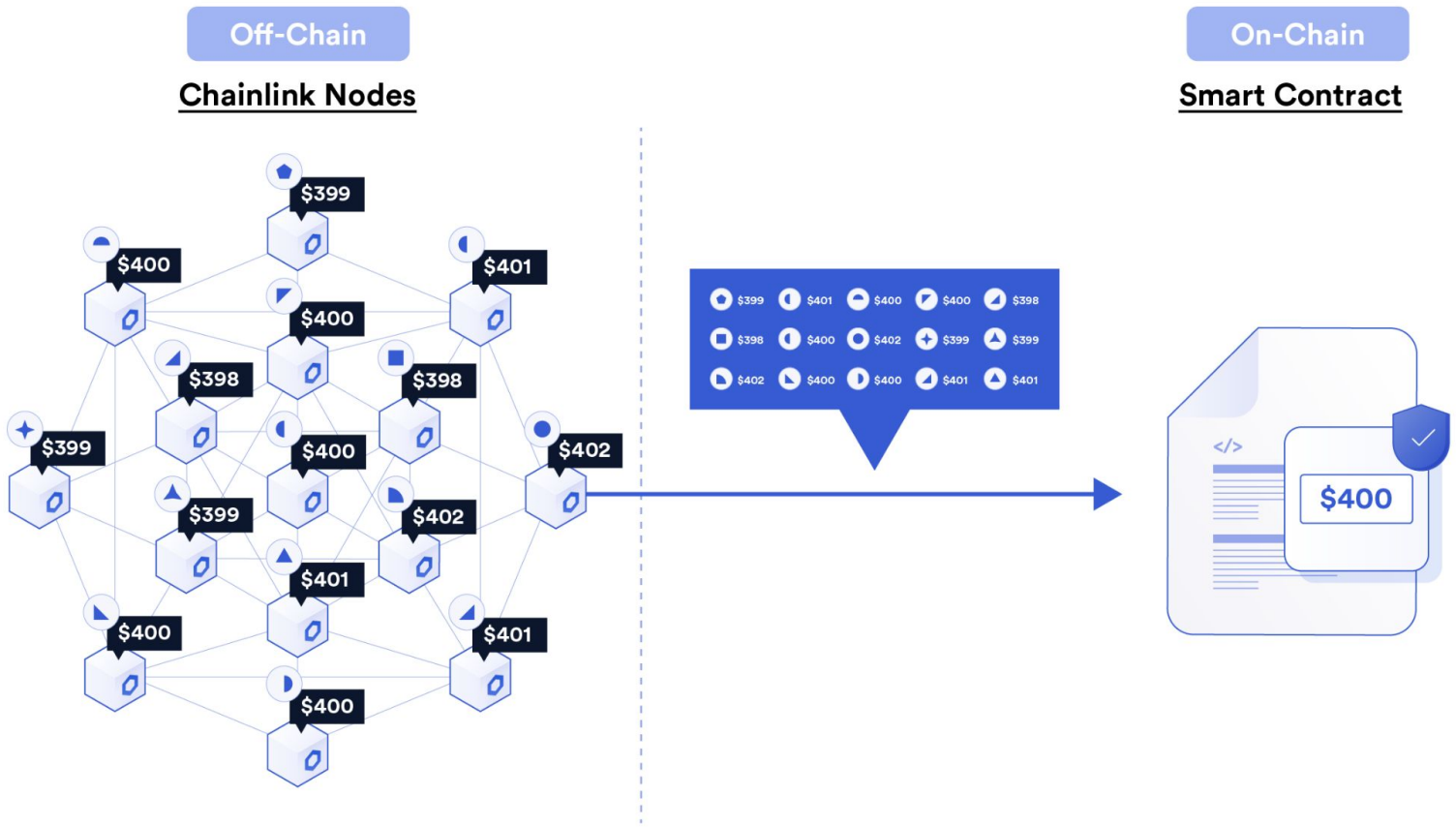
Full replicas being run by independent and sybil resistant node operators, coming to consensus about a computation.

Focused on data validation and consensus about individual off-chain values to make them reliable enough to trigger contracts.

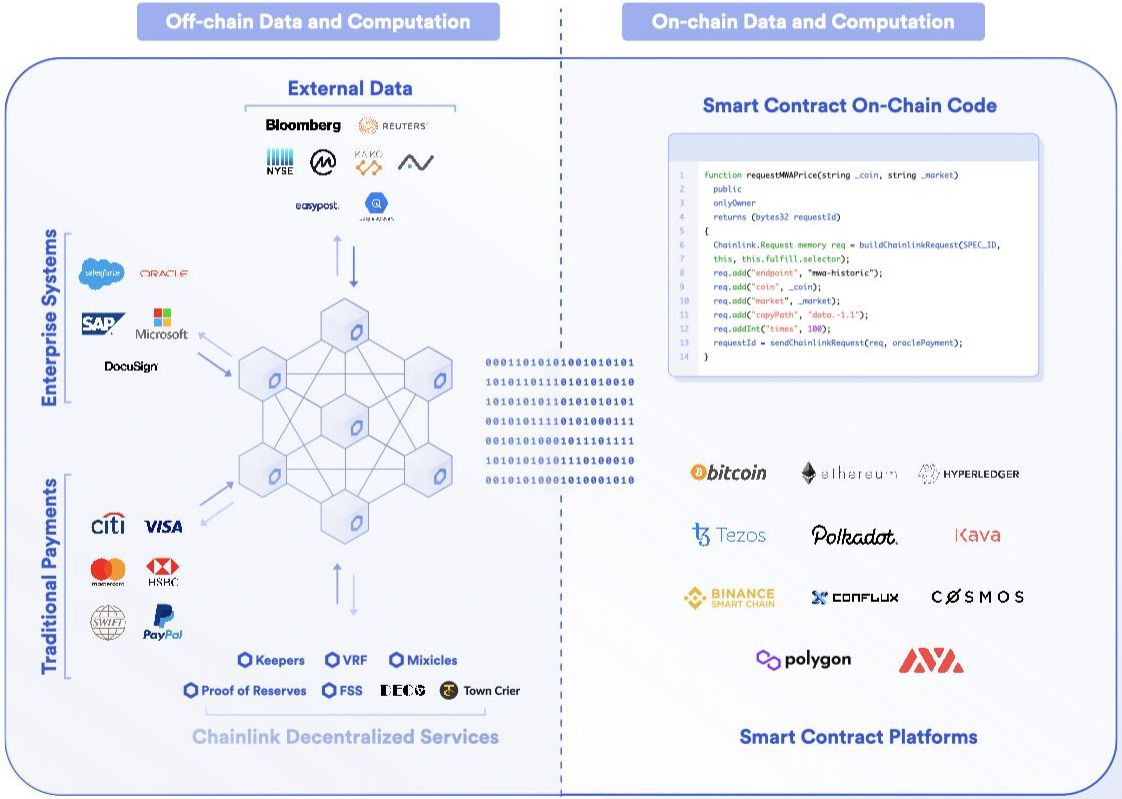
Node Operators are security reviewed, can provide a proven performance history and are high quality and highly sybil resistant.

End-to-end Reliability Is The Promise of Smart Contracts

Chainlink

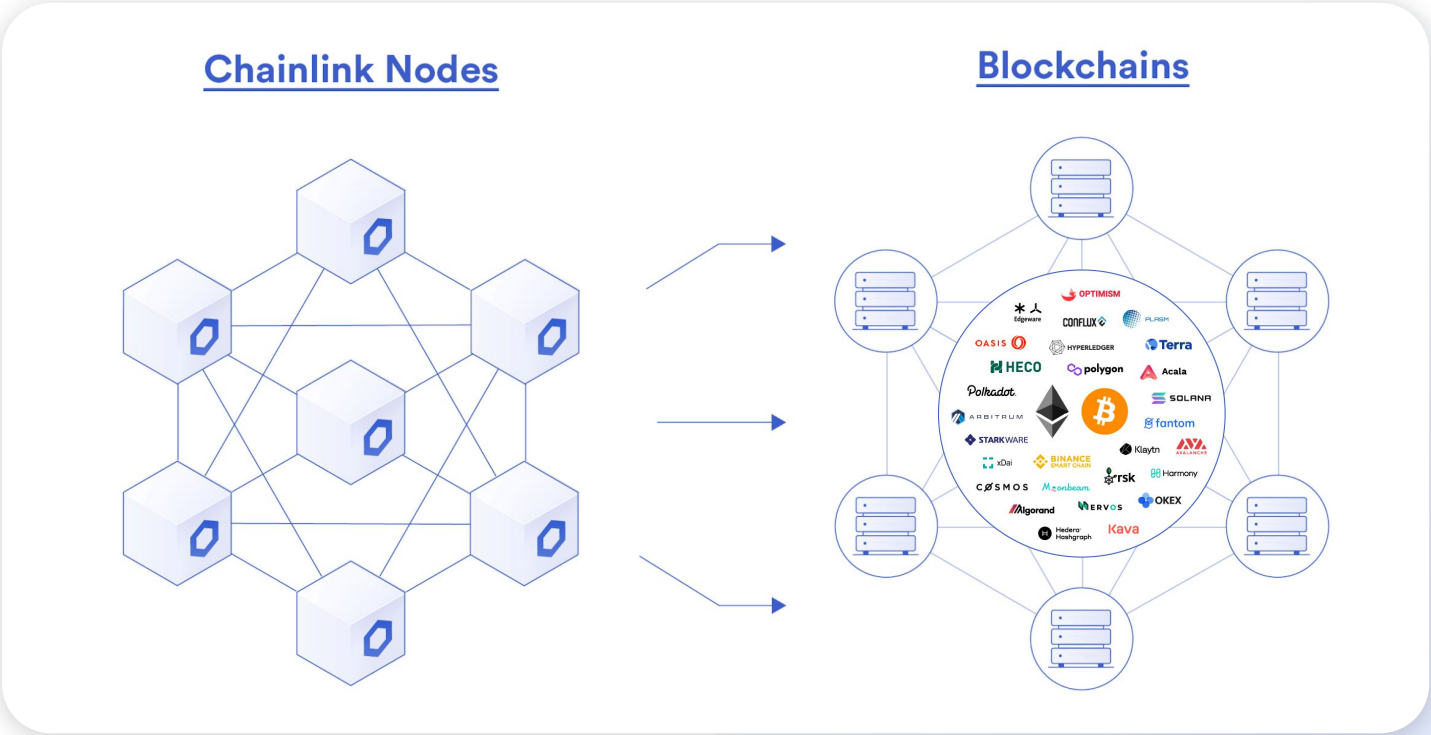

Off-Chain
Chainlink Nodes
On-Chain
Smart Contract
$399
$400
$401
$400
$398
$398
$399
$400
$402
$399
$402
$401
$400
$401
$400
$399
$401
$400
$400
$398
$398
$400
$402
$399
$399
$402
$400
$401
$401
$400

Hybrid Smart Contracts Combine On-chain Code & Off-chain Systems



Chainlink

Chainlink is Blockchain Agnostic



Chainlink

Chainlink Data Feeds

Chainlink

DeFi Completely Redefines a User's Relationship to Markets

Average Users

01011011

Decentralized Finance

01011011

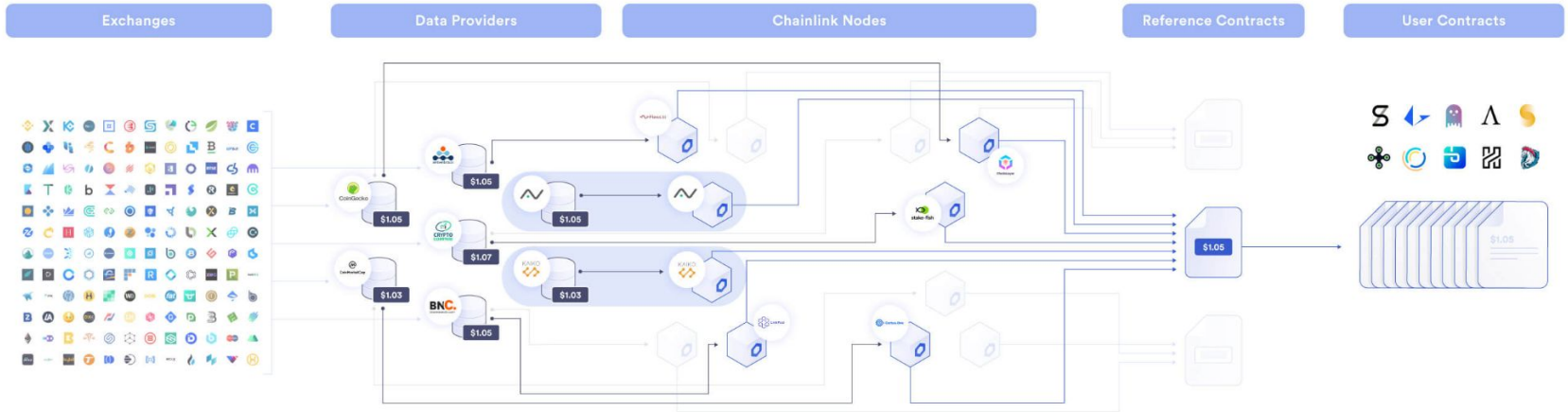
Global Financial Markets

Nasdaq

DeFi Trend: The Solution — and Huge Opportunity — to the Robinhood Fiasco

Chainlink

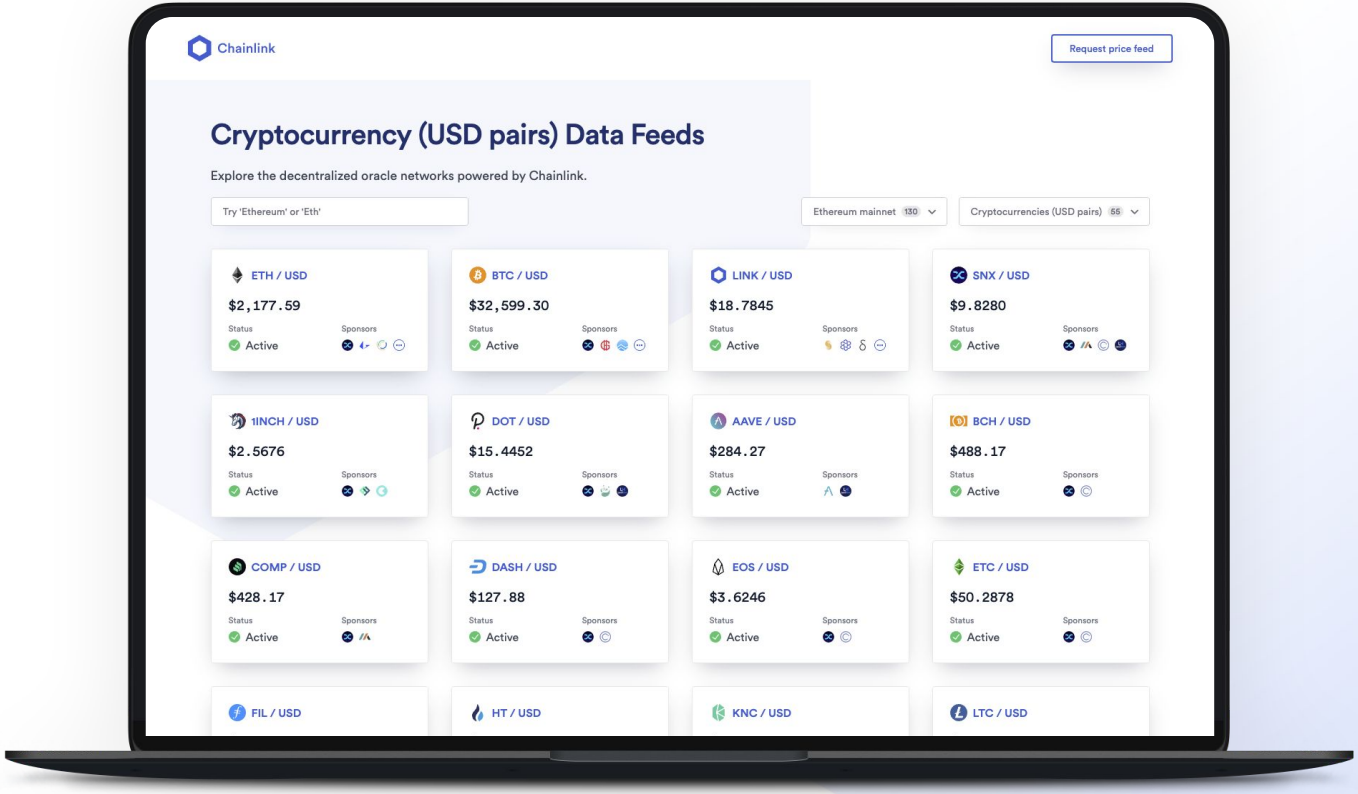
Price Data Enables DeFi to Reinvent the Financial System



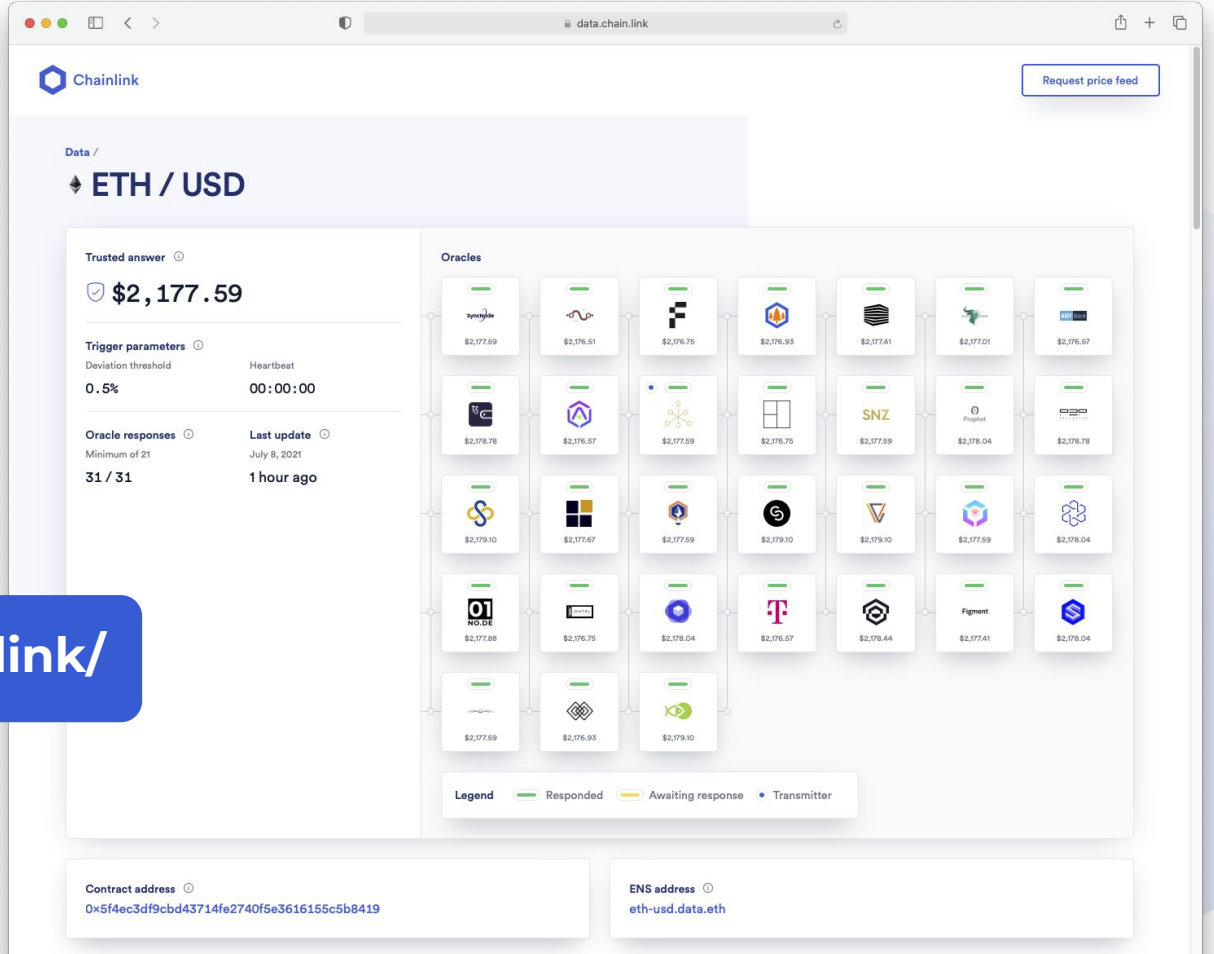
Chainlink

Chainlink

https://data.chain.link/

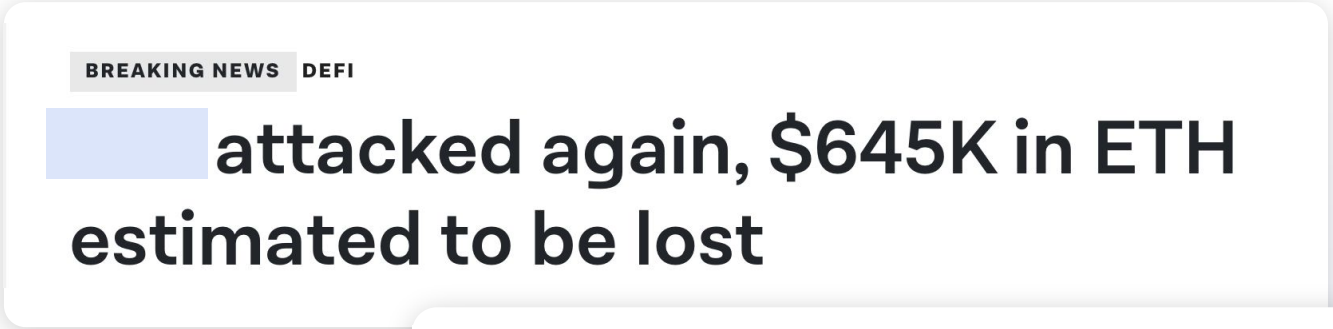


Data Feeds



https://data.chain.link/

We've already seen attacks



Chainlink

Demo 2: Chainlink Price Feeds

Exercise 3: Chainlink Price Feeds

- Remix: http://remix.ethereum.org/
- Chainlink Price Feeds Documentation: https://docs.chain.link/docs/using-chainlink-reference-contracts/

Chainlink

Accessing Any API Using Chainlink

Access Any API

Chainlink

Access Any API

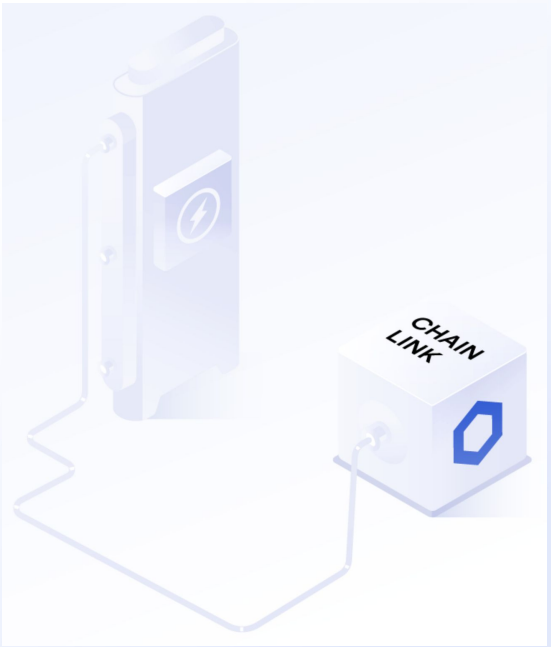
Chainlink

Chainlink Adapters

- Allow Chainlink Oracles to integrate with any external API
- Split into core and external adapters
- External adapters implemented a standardized API, with a structured input and output JSON specification

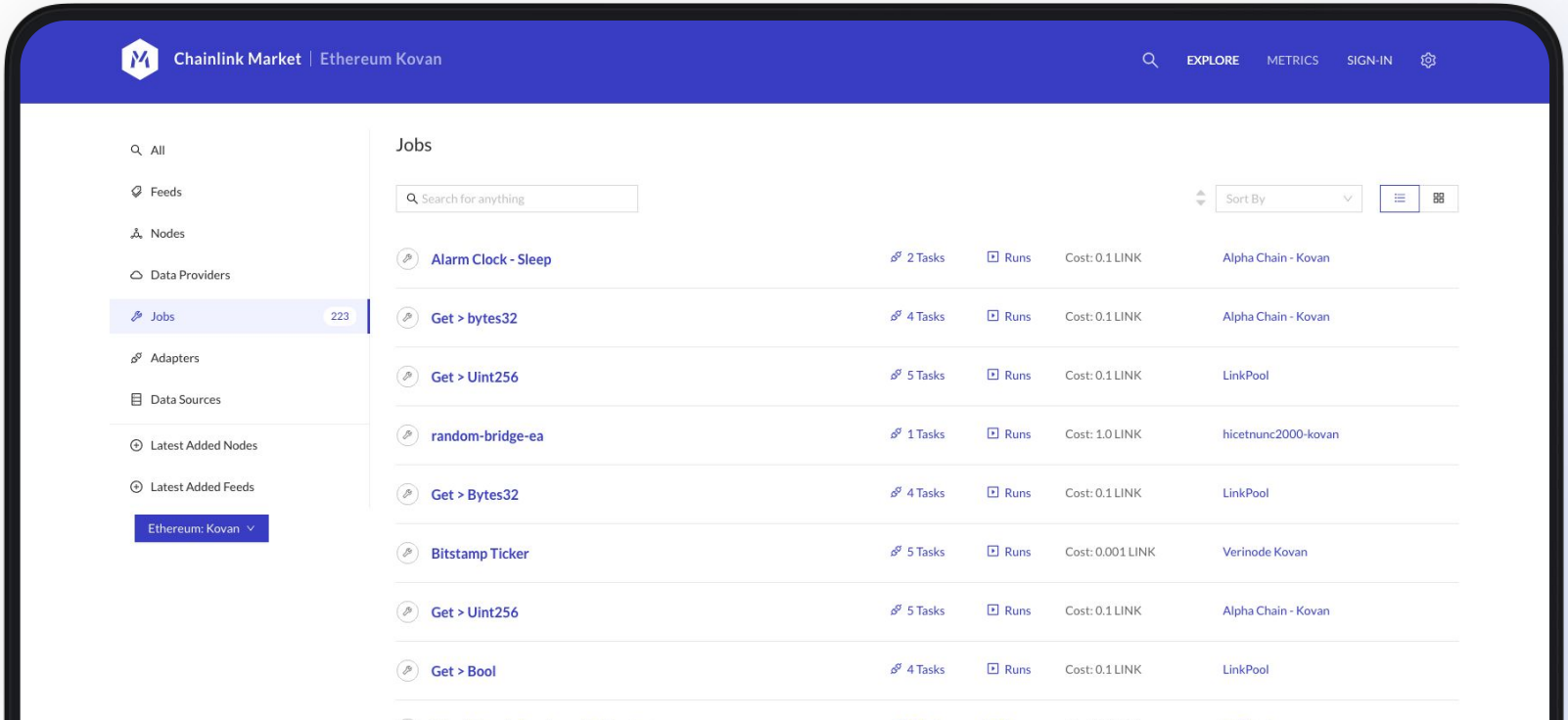
```
{ "id": 0, "data": { "from": "ETH", "to":
"USD" } }
```

```
{"jobRunID":0,"data":{"USD":441.49,"result":44
1.49},"result":441.49,"statusCode":200}
```



Chainlink Market

- Market for Chainlink jobs, adapters, data sources, feeds and data providers
- https://market.link/



Demo 4: Any-API

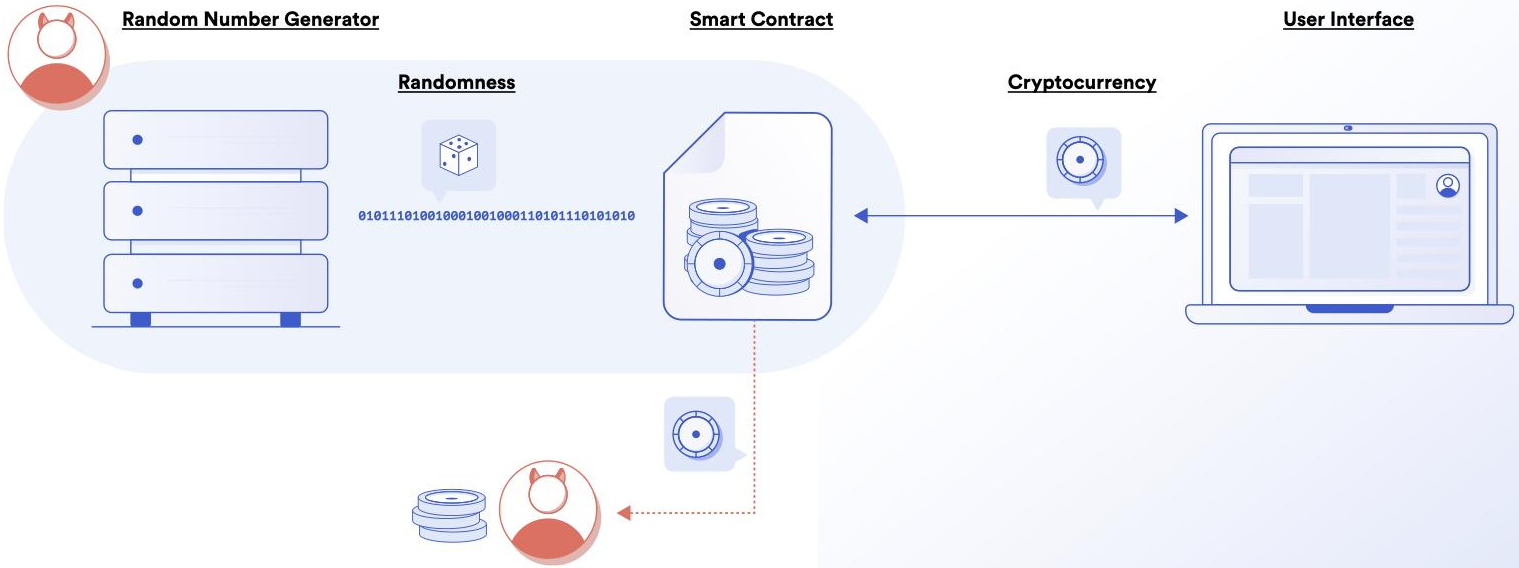
Exercise 4: Any-API

- Remix: http://remix.ethereum.org/
- Chainlink Any-API Documentation: https://docs.chain.link/docs/request-and-receive-data/

Chainlink

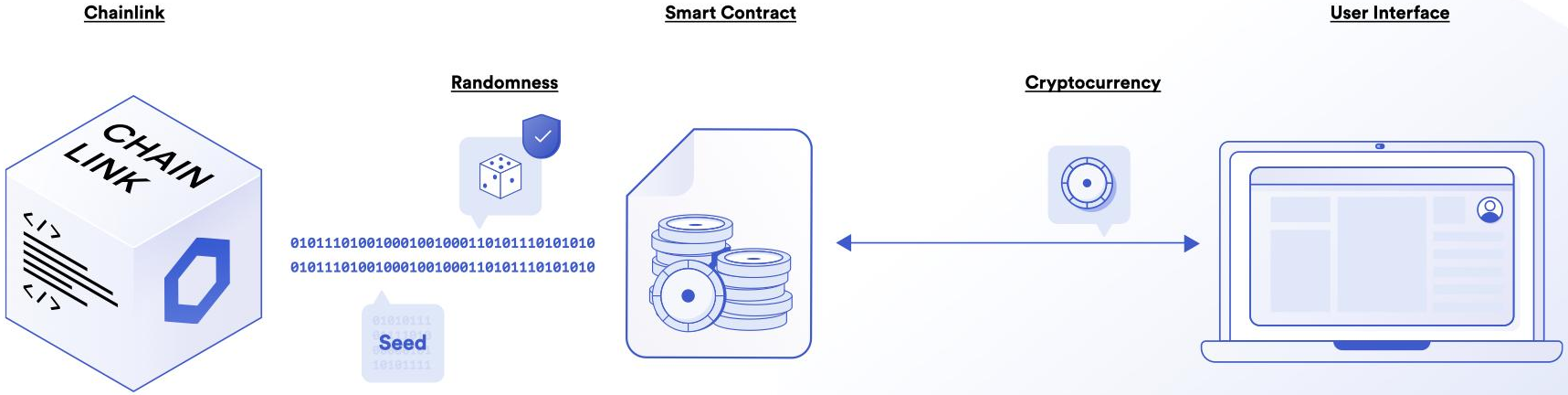
Chainlink VRF
(Verifiable Random Function)

Malicious RNG Operators are a Risk



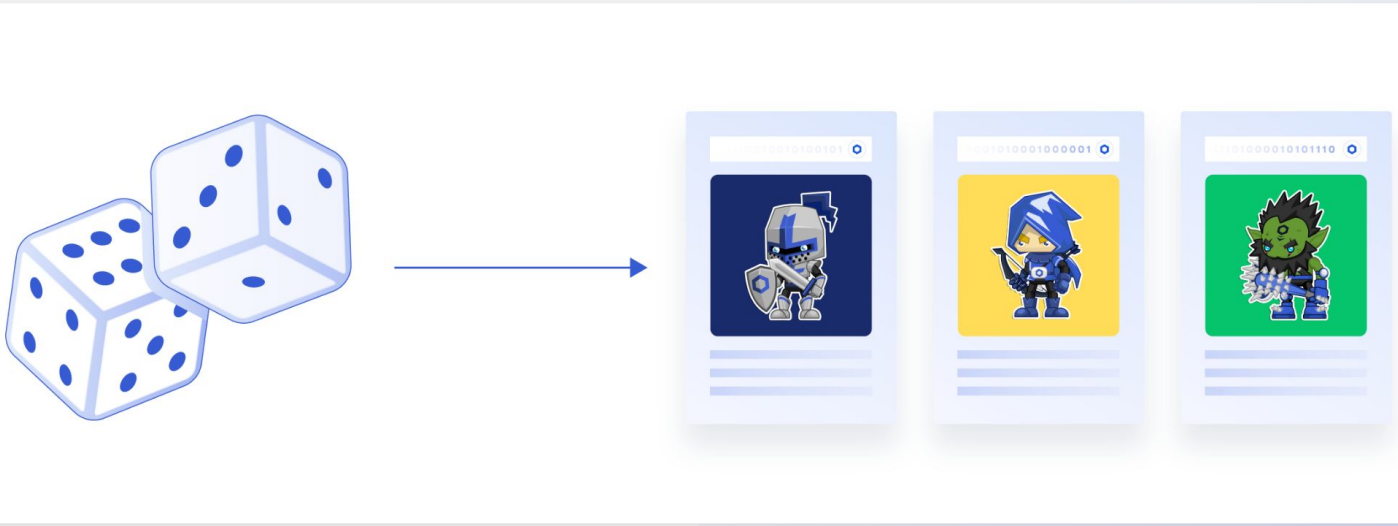
Chainlink

Chainlink VRF Provides Verifiable Randomness

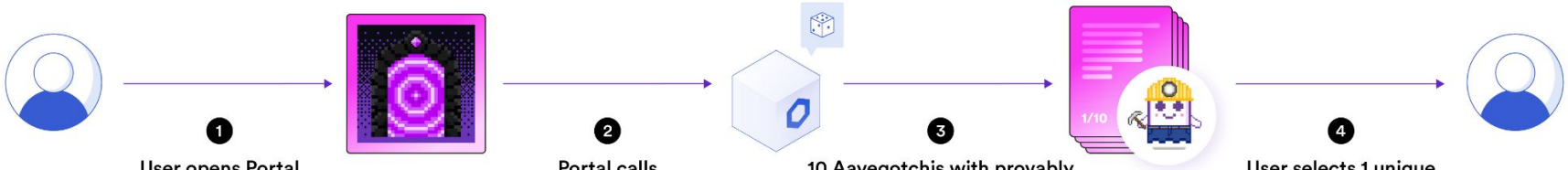


Chainlink

VRF Randomness used to generate provably random pack



Chainlink



Artists

Independent artists create unique digital artworks and content

EtherCards Framework

The EtherCards Platform provides tools for NFT creation

Chainlink VRF

Each NFT minted uses Chainlink VRF to assign provably rare traits

NFTs with randomized traits

Artists can now monetize their work with NFT rarity secured by Chainlink

Demo 5: Chainlink VRF

Exercise 5: Chainlink VRF

- Remix: http://remix.ethereum.org/
- Chainlink Price Feeds Documentation: https://docs.chain.link/docs/chainlink-vrf/

Developer Bootcamp Day 1 Summary

- Blockchains and smart contracts
- Ethereum and Solidity
- The oracle problem and Chainlink
- DeFi and Chainlink Data Feeds
- Accessing any API with Chainlink
- Chainlink VRF

Chainlink

Chainlink

Congratulations for completing Day 1 of the Smart Contract Developer Bootcamp