



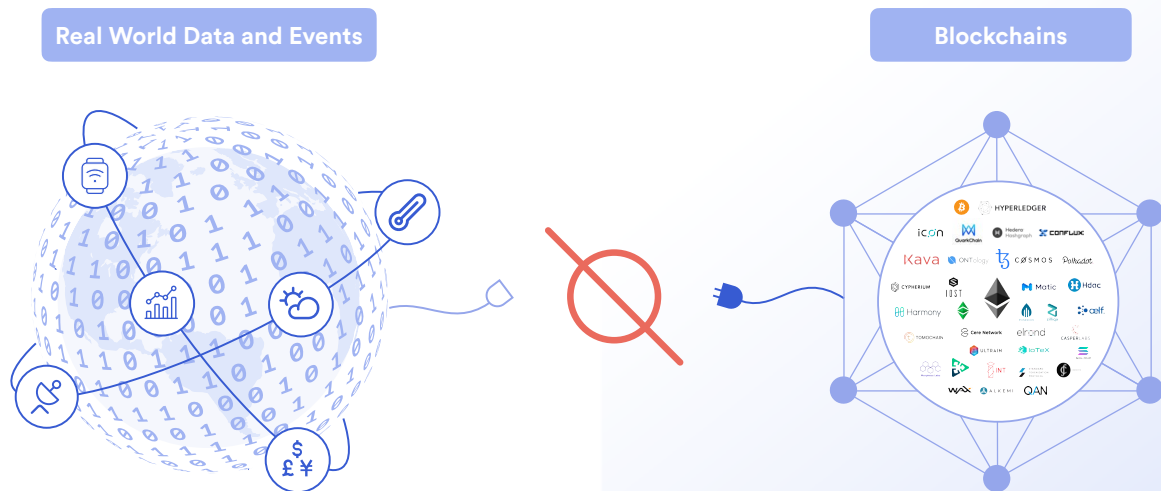
Chainlink

Externally Connected Smart Contracts

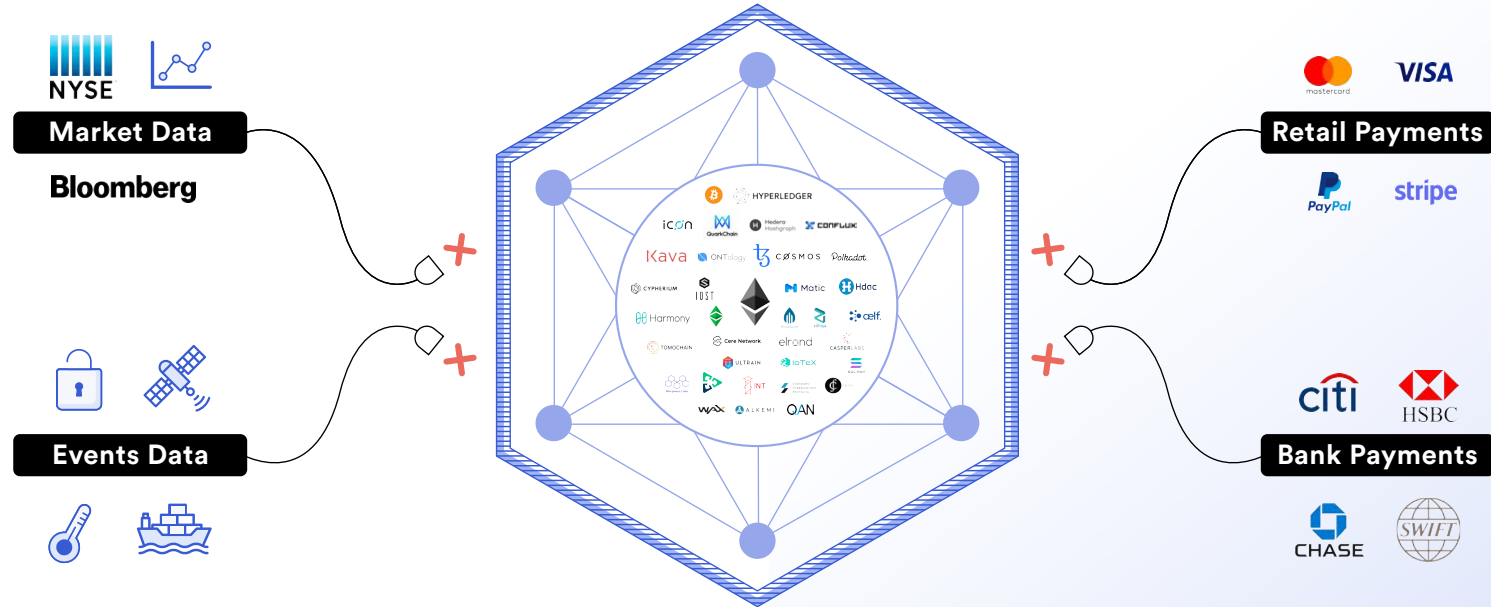
Consensus 2020

The “Oracle Problem” for Smart Contracts

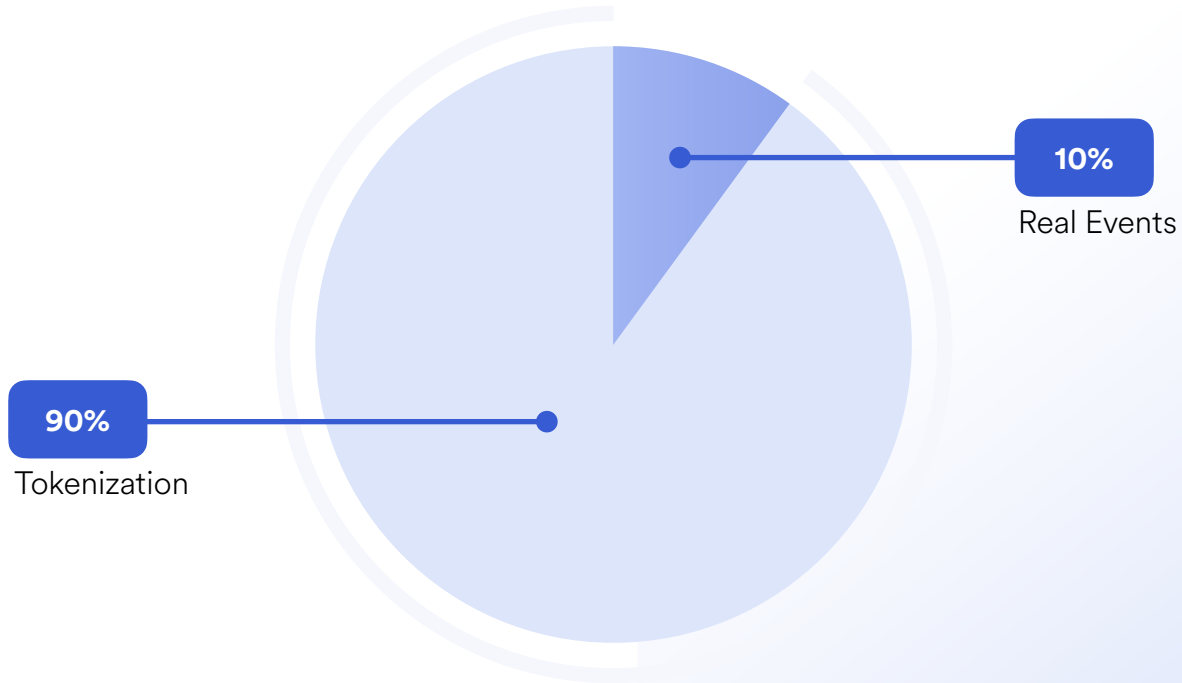
Smart Contracts are unable to connect with external systems, data feeds, APIs, existing payment systems or any other off-chain resources on their own.



The “Oracle Problem” for Smart Contracts

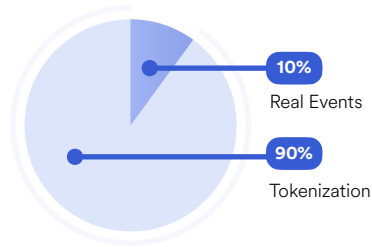


Smart Contracts are Currently Used for Tokenization



Redefining Smart Contracts With External Events

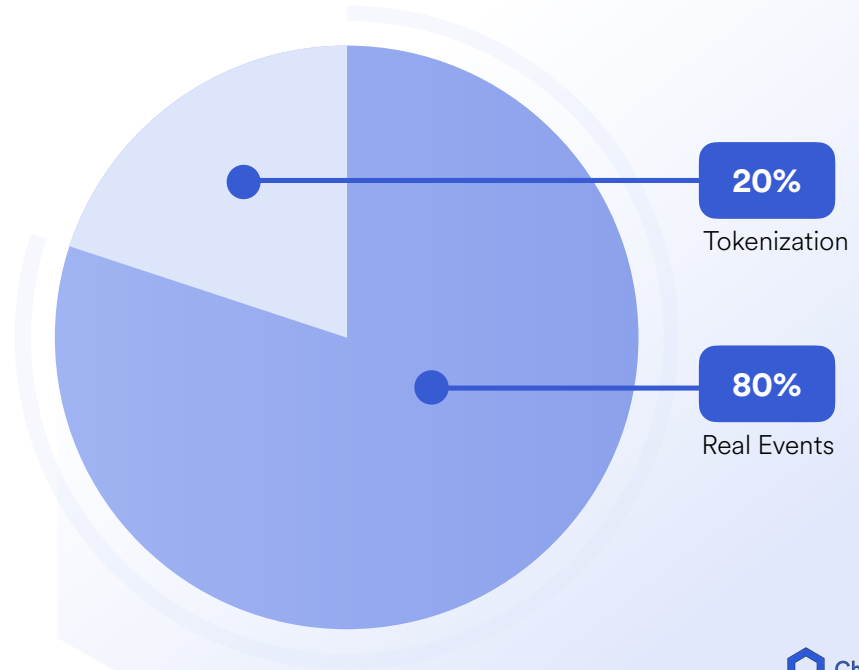
Current Distribution of Smart Contract
Transaction Volume and Value Secured



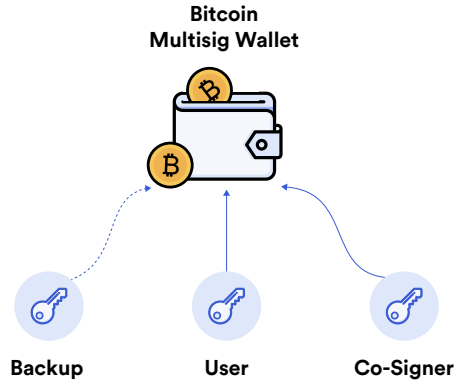
Externally
Connected
Contracts



Future Distribution and 1000%+ Growth
of Transaction Volume and Value Secured



The Initial Leap Forward for Smart Contracts



**Bitcoin Multi-signature as
“Programmable Money”**



**Protocol Smart Contracts =
Smart Contracts 1.0**

The Scriptable Leap Forward for Smart Contracts



**Protocol Smart Contracts =
Smart Contracts 1.0**



```
1 pragma solidity ^0.4.16;
2
3 contract MyToken {
4     // This creates an array with all balances
5     mapping (address => uint256) public balanceOf;
6
7     // Initializes contract with initial supply tokens to
8     function MyToken (
9         uint256 initialSupply
10    ) payable {
11         balanceOf[msg.sender] = initialSupply;
12     }
13
14     // Send coins
15     function transfer(address _to, uint256 _value) payable
16     {
17         require(balanceOf[msg.sender] >= _value);
18         require(balanceOf[_to] + _value >= balanceOf[_to]);
19         balanceOf[msg.sender] -= _value;
20         balanceOf[_to] += _value;
21     }
22 }
```

**Scriptable Smart Contracts =
Tokenization/Smart Contracts 2.0**

The Connectivity Leap Forward for Smart Contracts

```
1 pragma solidity ^0.4.16;
2
3 contract MyToken {
4     // This creates an array with all balances
5     mapping (address => uint256) public balanceOf;
6
7     // Initializes contract with initial supply tokens to
8     function MyToken (
9         uint256 initialSupply
10     ) payable {
11         balanceOf[msg.sender] = initialSupply;
12     }
13
14     // Send coins
15     function transfer(address _to, uint256 _value) payable
16     {
17         require(balanceOf[msg.sender] >= _value);
18         require(balanceOf[_to] + _value >= balanceOf[_to]);
19         balanceOf[msg.sender] -= _value;
20         balanceOf[_to] += _value;
21     }
22 }
```



**Scriptable Smart Contracts =
Tokenization/Smart Contracts 2.0**

**Externally Connected Smart Contracts =
All Other Dapps/Smart Contracts 3.0**

DeFi is the Beginning of Redefining Smart Contracts

Total Value Locked (USD) in DeFi

TVL (USD)



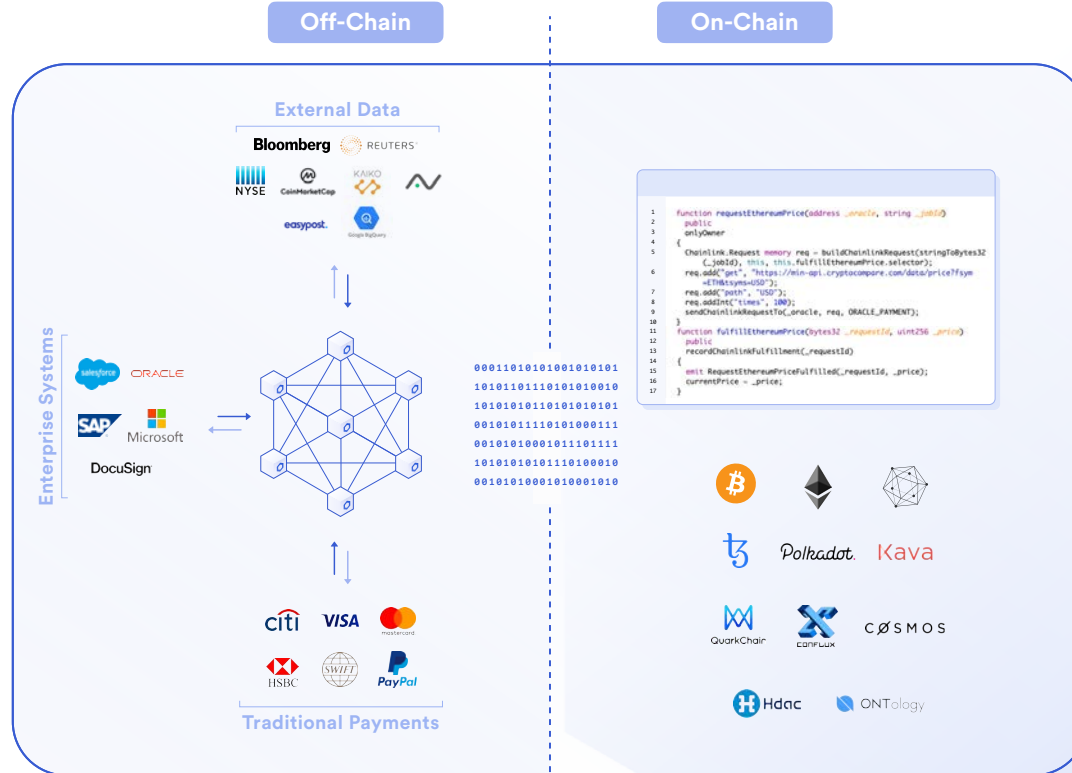
SYNTHETIX

AAVE

bZx

Nexus Mutual

DeFi Smart Contracts Have Two Important Parts



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

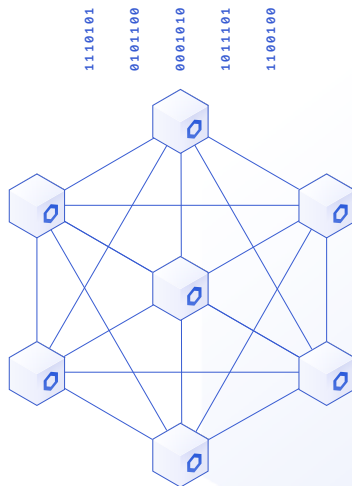
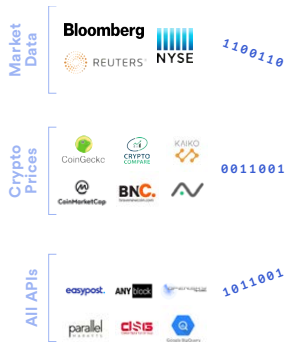


Connecting Smart Contracts to All Inputs and Outputs

Smart Contracts
& Blockchains



Contract Inputs



Contract Outputs



Our Approach to Reliable & Secure Oracles for Web3



Decentralization of many
Independent/Sybil Resistant
Nodes into Oracle Networks

Our Approach to Reliable & Secure Oracles for Web3



Decentralization of many
Independent/Sybil Resistant
Nodes into Oracle Networks



Provably Secure Nodes
that provide cryptographic
proof of their overall security

Our Approach to Reliable & Secure Oracles for Web3



Decentralization of many
Independent/Sybil Resistant
Nodes into Oracle Networks



Provably Secure Nodes
that provide cryptographic
proof of their overall security



High Quality Data
from multiple reliable sources and
well validated by multiple nodes

Our Approach to Reliable & Secure Oracles for Web3



Decentralization of many
Independent/Sybil Resistant
Nodes into Oracle Networks



Provably Secure Nodes
that provide cryptographic
proof of their overall security



High Quality Data
from multiple reliable sources and
well validated by multiple nodes



Cryptoeconomic Security
using binding service agreements
to generate staking penalties

Our Approach to Reliable & Secure Oracles for Web3



Decentralization of many Independent/Sybil Resistant Nodes into Oracle Networks



Provably Secure Nodes that provide cryptographic proof of their overall security



High Quality Data from multiple reliable sources and well validated by multiple nodes



Cryptoeconomic Security using binding service agreements to generate staking penalties



Defense in Depth, applying multiple layers of security (TEEs, ZK)

Our Approach to Reliable & Secure Oracles for Web3



Decentralization of many Independent/Sybil Resistant Nodes into Oracle Networks



Provably Secure Nodes that provide cryptographic proof of their overall security



High Quality Data from multiple reliable sources and well validated by multiple nodes



Cryptoeconomic Security using binding service agreements to generate staking penalties



Defense in Depth, applying multiple layers of security (TEEs, ZK)



A Large Open Source Community of Node Operators, Developers, Researchers and Security Auditors

Our Approach to Reliable & Secure Oracles for Web3



Decentralization of many Independent/Sybil Resistant Nodes into Oracle Networks



Provably Secure Nodes that provide cryptographic proof of their overall security



High Quality Data from multiple reliable sources and well validated by multiple nodes



Cryptoeconomic Security using binding service agreements to generate staking penalties



Defense in Depth, applying multiple layers of security (TEEs, ZK)



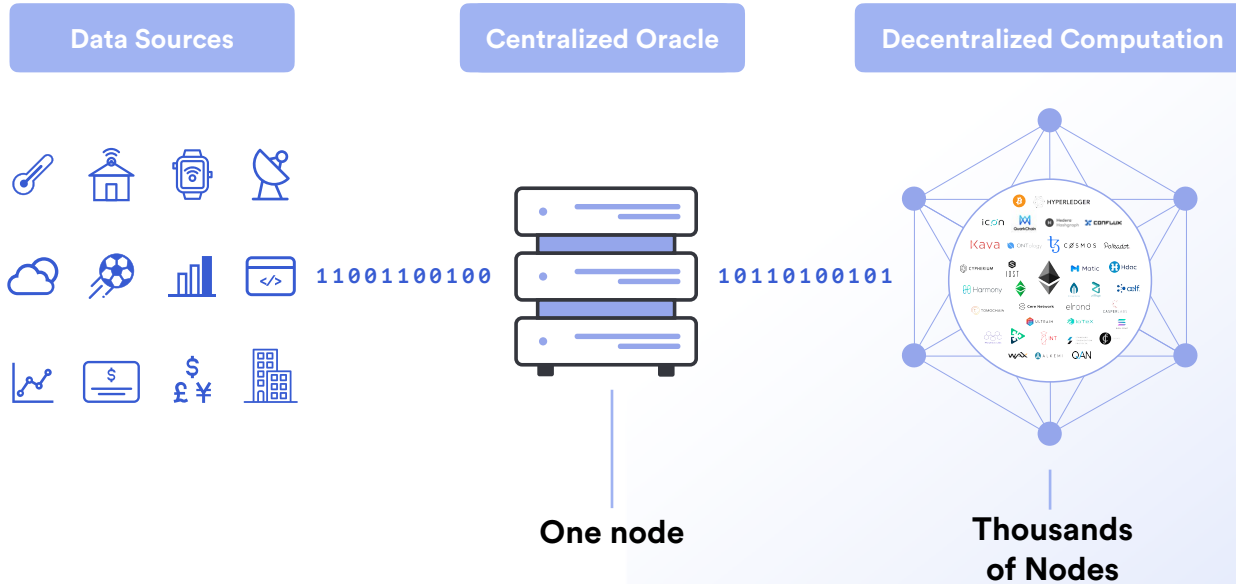
A Large Open Source Community of Node Operators, Developers, Researchers and Security Auditors



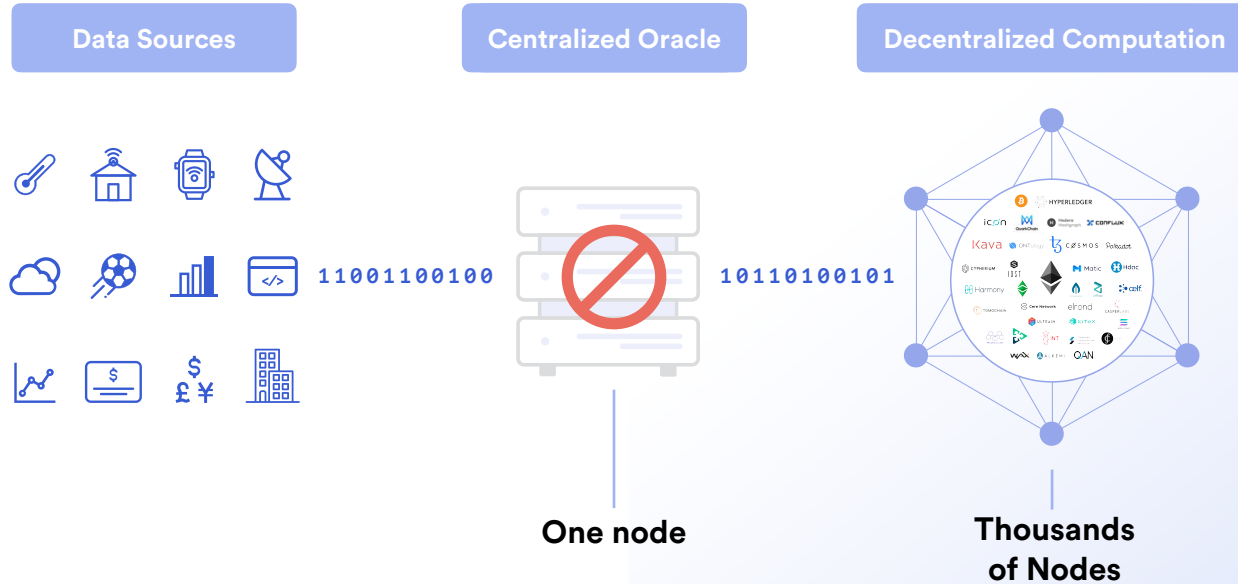
Connecting any blockchain environment to all inputs and outputs

Connected to the leading public and private blockchains. Accelerating what developers can build and the amount of places data can be sold on-chain.

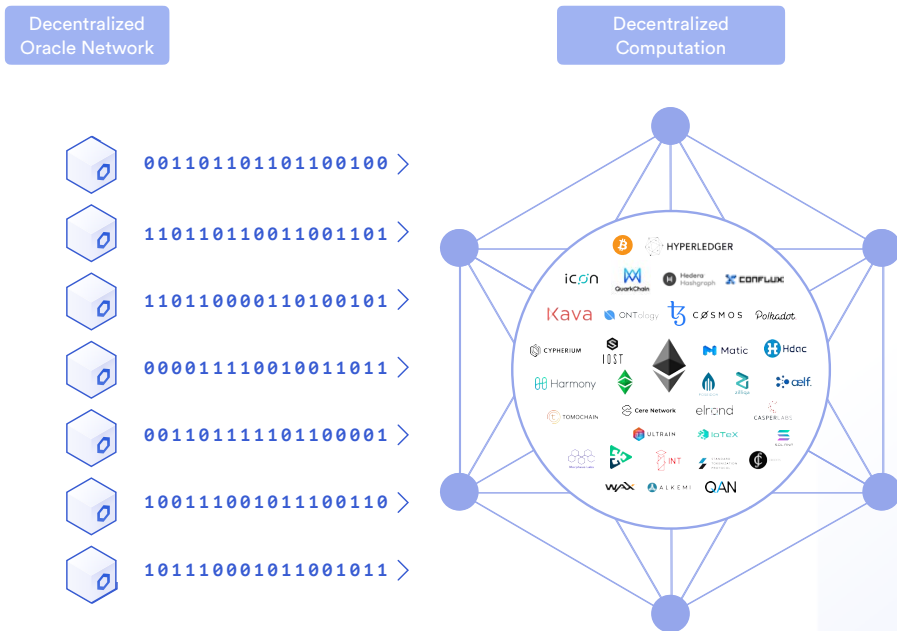
Centralized Oracles are a Point of Failure



Centralized Oracles are a Point of Failure



A Decentralized Oracle Network



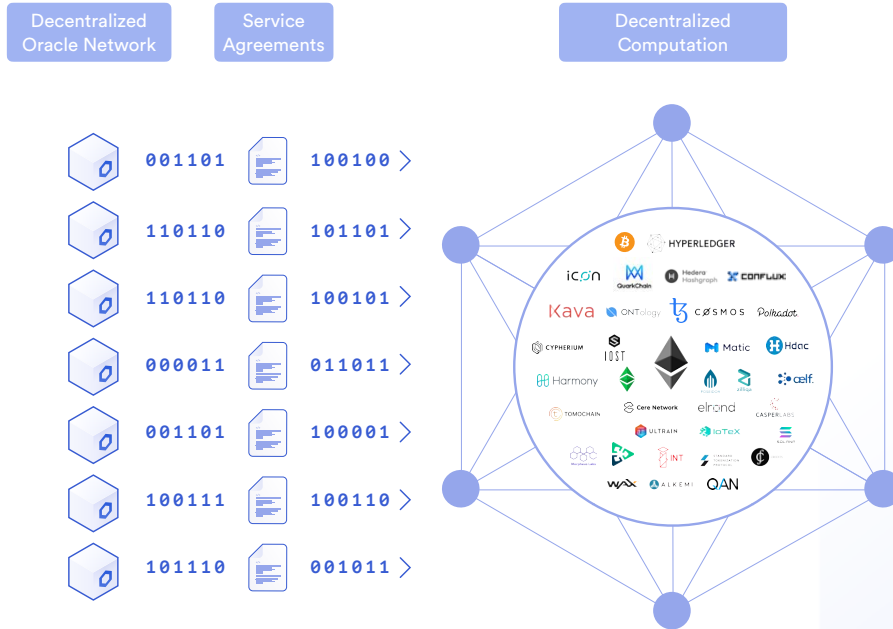
Decentralization

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.

Binding Commitments by Oracles to Contracts



Binding Service Agreements

Technically enforced commitments to meeting high security and data quality standards are made on-chain by the oracle, committing them to high levels of quality.

Both the commitment and the final performance of the commitment are both on-chain and fully verifiable.

Creating a cryptographically provable performance history that can be relied on. Oracles that don't fulfill their commitments won't be selected for future quorums, losing large future revenue.

On-chain Service Agreements Provide Guarantees

Data Providers



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

1010110111

Chainlink Node



1010110111

On-chain Service Agreement



10010100101001010010

- ✓ Defined Data Delivery Parameters
- ✓ Defined Data Quality Parameters
- ✓ Quality Connected to Payment

Smart Contract with High Quality Data



On-chain Service Agreements Provide Guarantees

Data Providers



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

1010111011

Chainlink Node



1010111011

On-chain Service Agreement



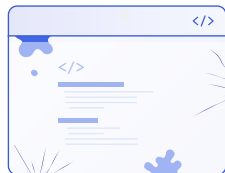
10010100101001010010

- ✓ Defined Data Delivery Parameters
- ✓ Defined Data Quality Parameters
- ✓ Data Quality Connected to Payment

Smart Contract with High Quality Data



Unpaid/Free OpenAPI



- ✗ Low Quality Data
- ✗ Unreliable Responses
- ✗ No Quality Guarantees

1010111010101111011

A Node Without Credential Management



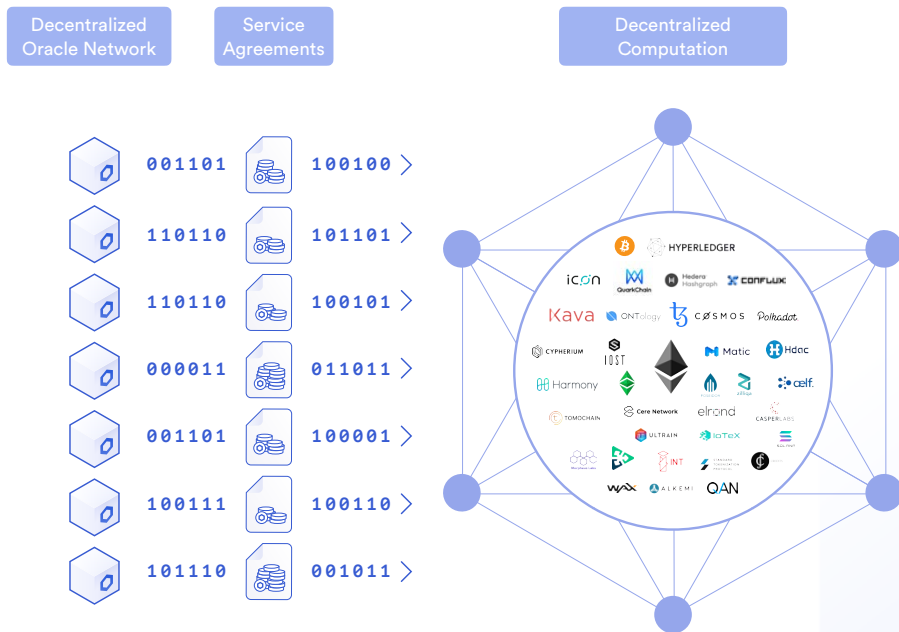
- ✗ Undefined Data Delivery Terms
- ✗ Undefined Data Quality Terms
- ✗ No Data Quality Guarantees

1010110010
1
0
0
0

Smart Contract With No Data Delivered



Crypto-economic Security from Staking



Cryptoeconomic Security from Staking

Using the commitments from binding service agreements, we can define clear parameters for the penalties for misbehaving nodes, bad data providers and any other point in the data origination and/or data transfer process.

Staking ensures that there is a penalty for node misbehavior, data inaccuracy and any other key condition specified in an on-chain binding service agreement. Staking guarantees are not only very specific and expandable, to properly manage new risks as they appear, but the amount of stake can be increased as value secured by an oracle rises.

Crypto-economic Security from Staking

Data Providers



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

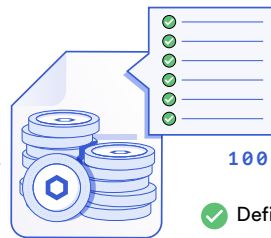
1010110111

Chainlink Node



1010110111

Service Agreement



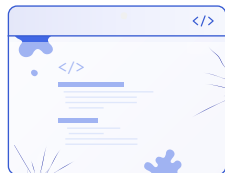
10010100101001010010

- ✓ Defined Data Delivery Parameters
- ✓ Defined Data Quality Parameters
- ✓ Quality Connected to Payment

Smart Contract with High Quality Data



Unpaid/Free OpenAPI



- ✗ Low Quality Data
- ✗ Unreliable Responses
- ✗ No Quality Guarantees

1010111010101111011

A Node Without Credential Management



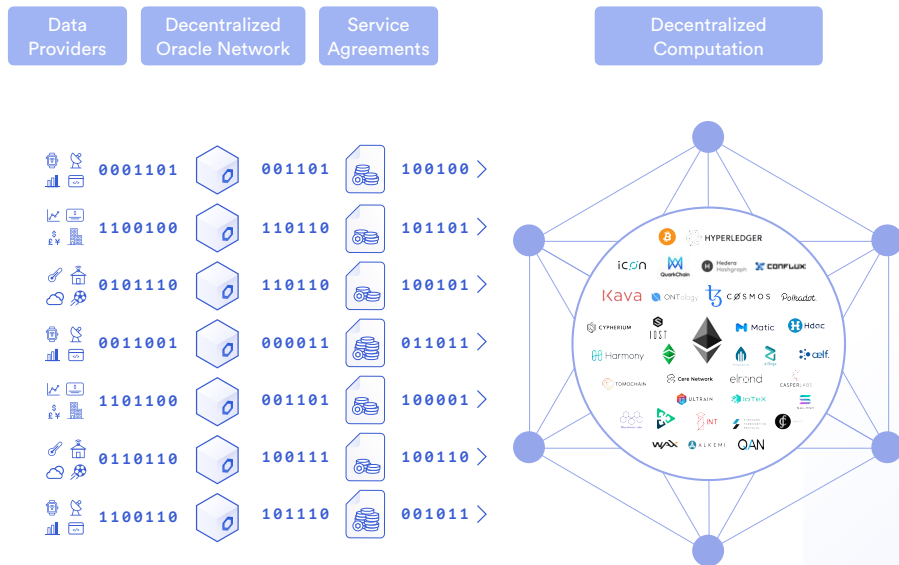
- ✗ Undefined Data Delivery Terms
- ✗ Undefined Data Quality Terms
- ✗ No Data Quality Guarantees

1010110010
0
1
0
0
0

Smart Contract Broken by Bad Data



Decentralization at the Data Source Level

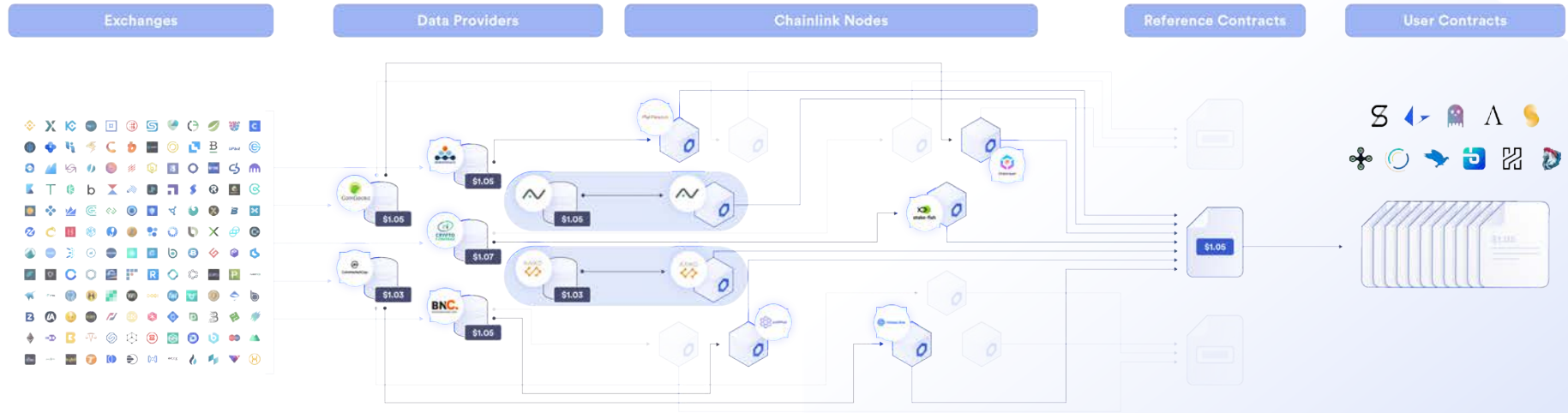


Highly Validated Data

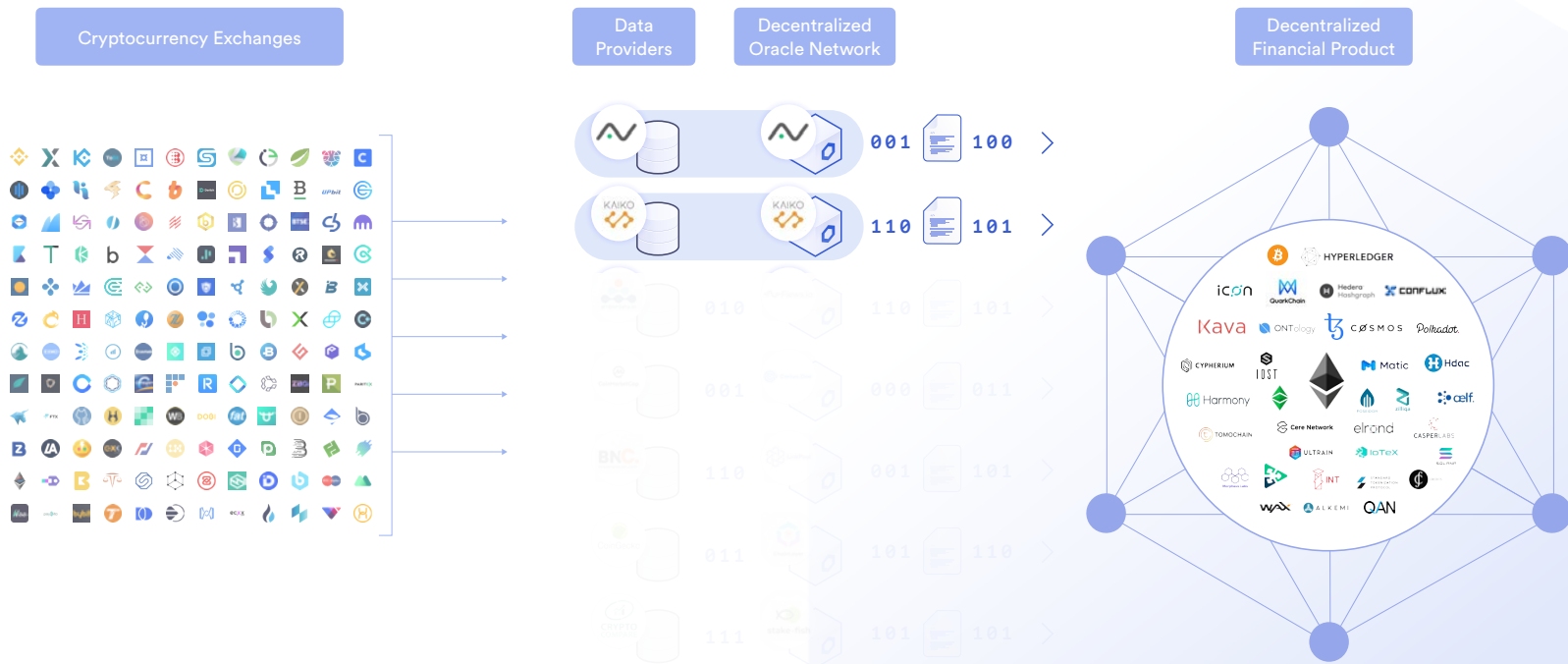
Decentralization at the middleware layer enables the inclusion of multiple secured data sources

Pre-made Chainlinks make it easy to build multiple data sources into your smart contract, with data quality grantees and data delivery grantees from node operators built in.

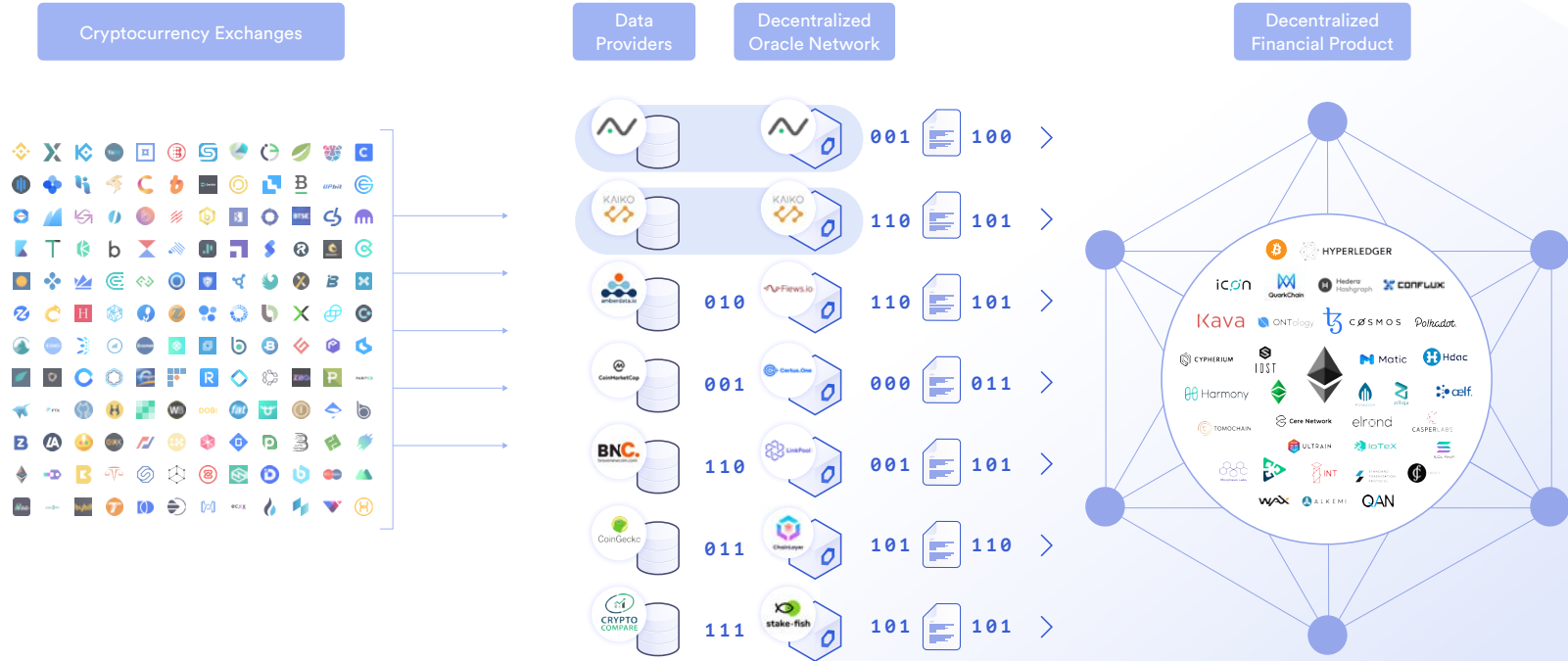
Truly Decentralized Finance via Decentralized Oracles



Truly Decentralized Finance via Decentralized Oracles



Truly Decentralized Finance via Decentralized Oracles





Chainlink

[Back to listing](#)

0xF5fff180082d6017036B771bA883025c654BC935

BTC / USD aggregation

Latest and trusted answer

\$ 9274.114

Primary Aggregation Parameter

Deviation Threshold: 1%

Secondary Aggregation Parameter

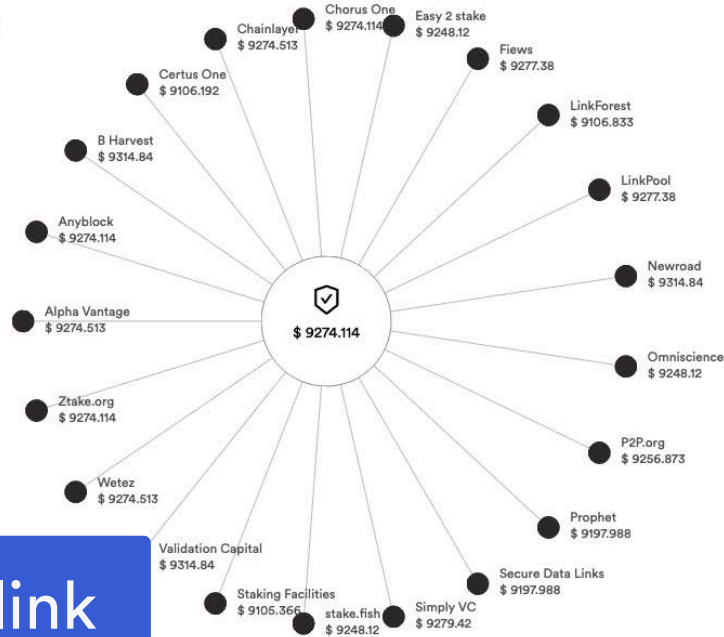
Heartbeat: 01:43:20

Oracle responses (minimum 14)

21 / 21

Update date May 6th 2020

8:01 PM



<https://feeds.chain.link>

<https://feeds.chain.link/btc-usd>

Introducing Chainlink VRF

0	5	8	2	5	8	9	7
5	6	7	4	1	2	3	0
7	4	7	0	3	4	7	4
4	8	0	4	5	0	4	4
5	6	2	3	5	4	8	7
5	9	8	7	4	6	1	2

On-chain Verifiable Randomness for Smart Contract Developers



**Provably
Fair**



**Provably
Random**



**Cost
Effective**

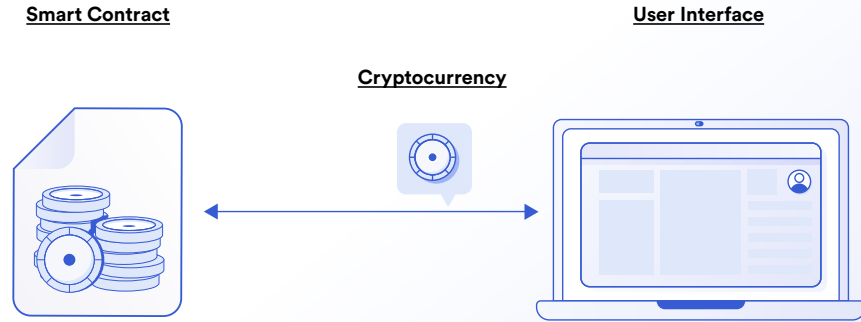


**Reliable
and Secure**



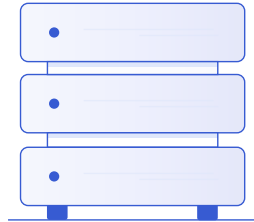
**Easy to
Integrate**

Blockchain Gaming's Use of Randomness



Blockchain Gaming's Use of Randomness

Random Number Generator



Randomness



010111010010001001000110101110101010

Smart Contract



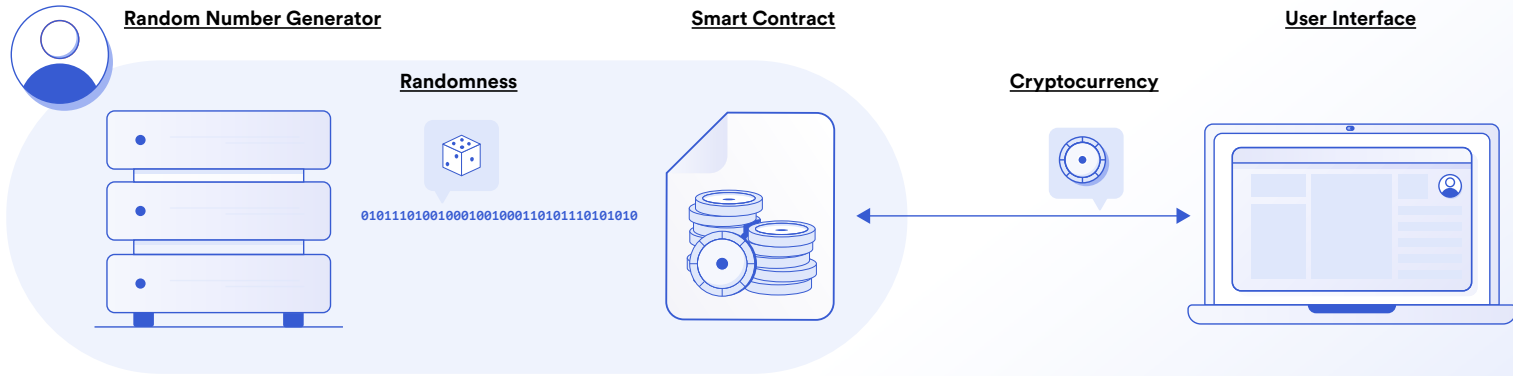
Cryptocurrency



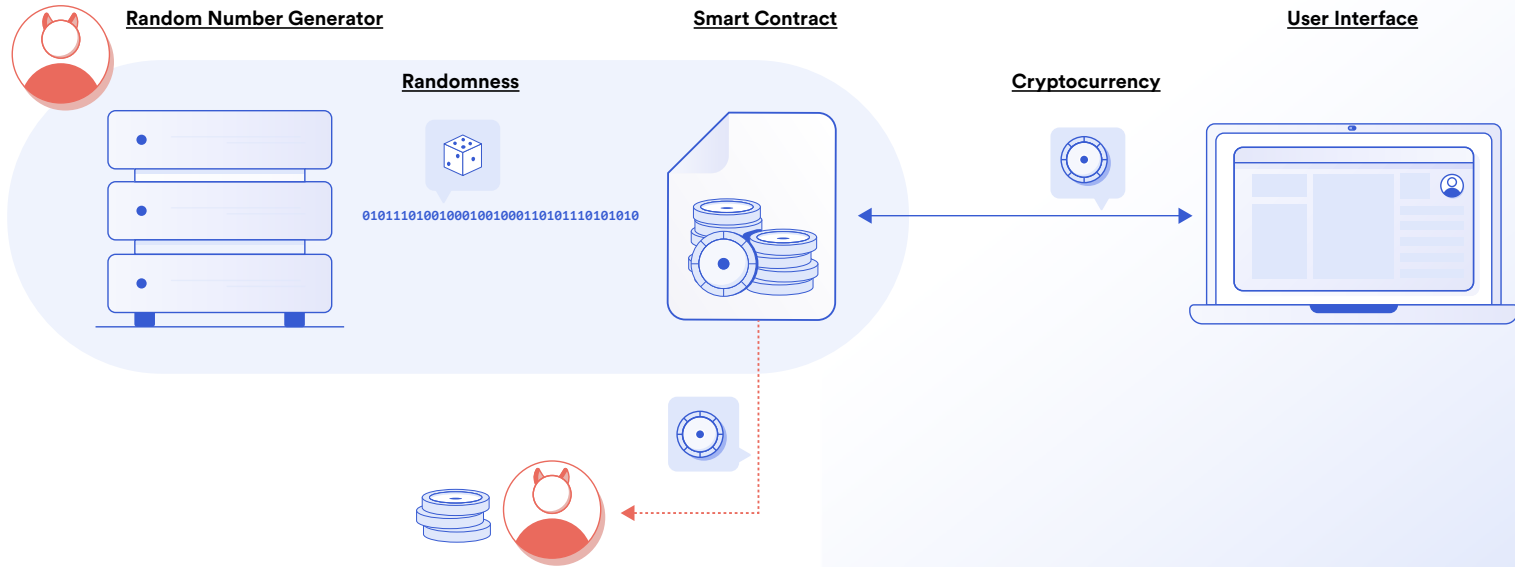
User Interface



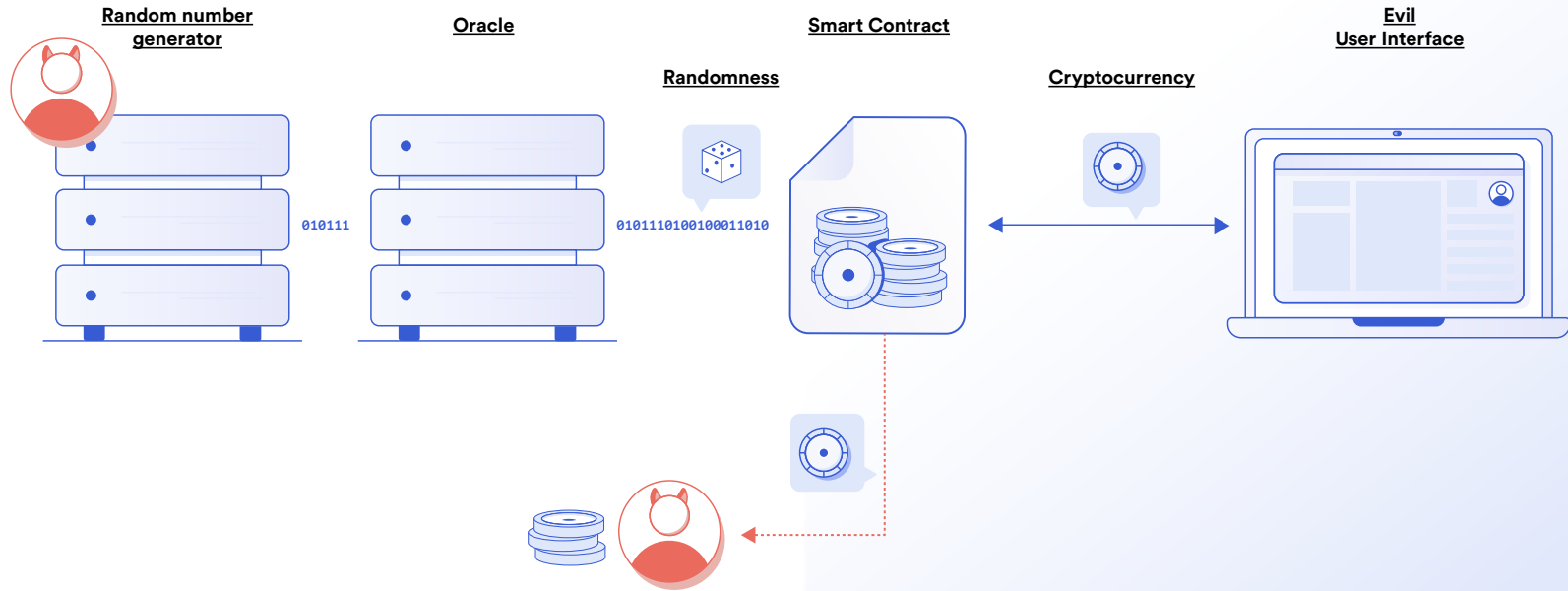
Blockchain Gaming's Use of Randomness



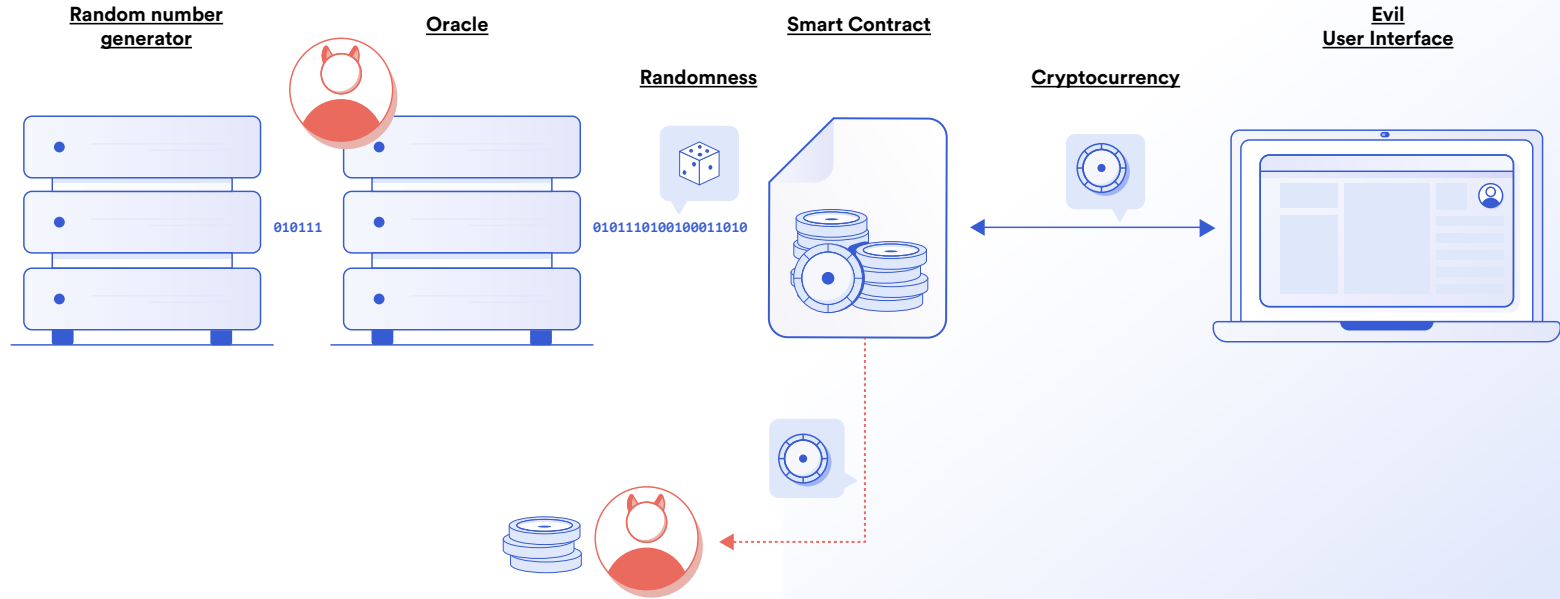
Malicious RNG Operators are a Risk



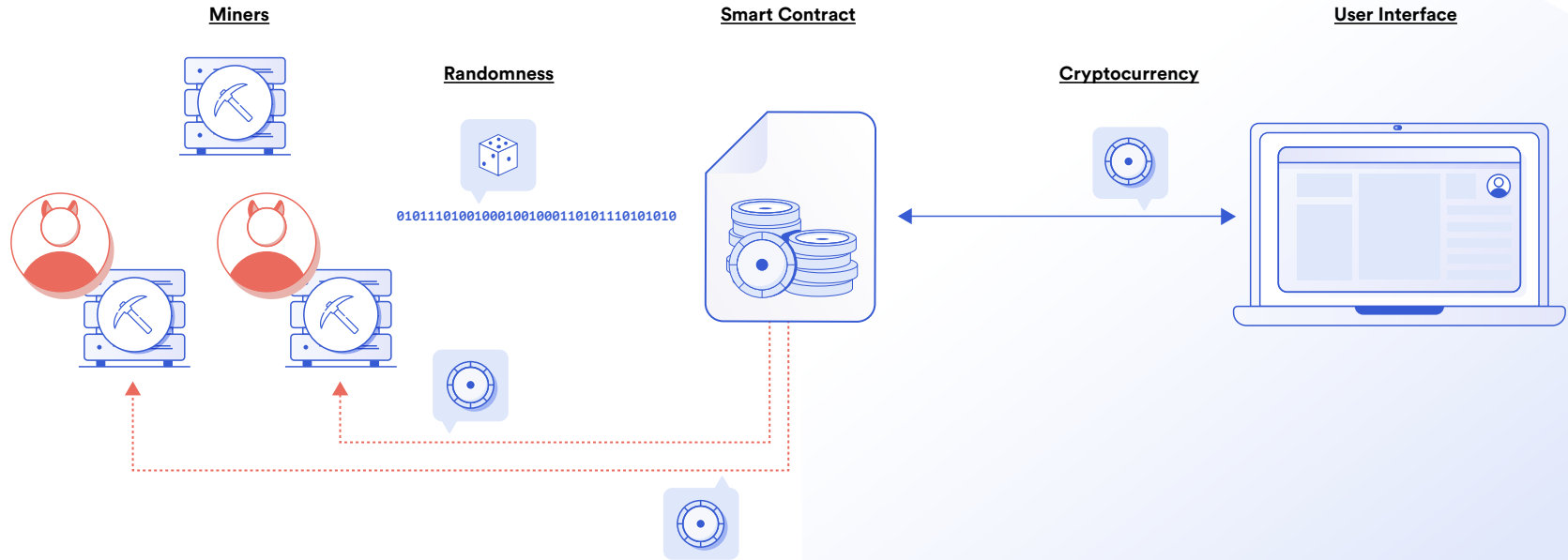
Malicious RNG Operators are a Risk



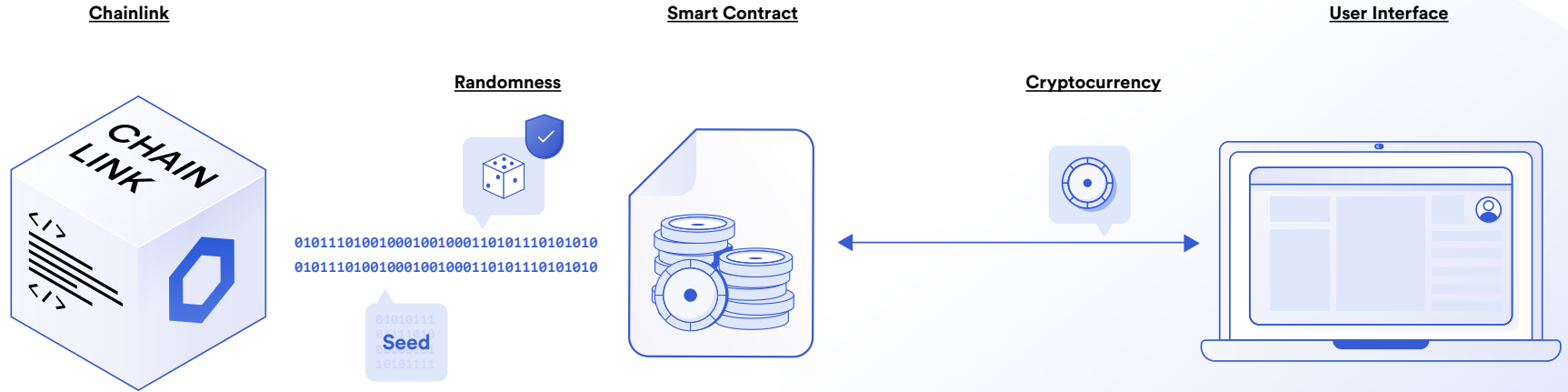
Single Malicious Oracle Operators are a Risk



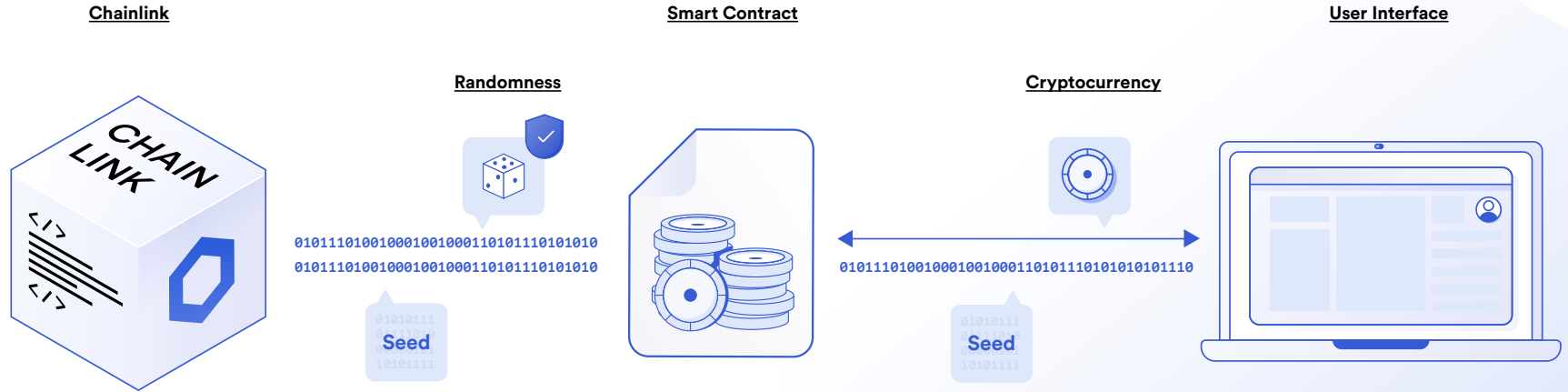
Malicious Miners are a Risk when using Blockhashes



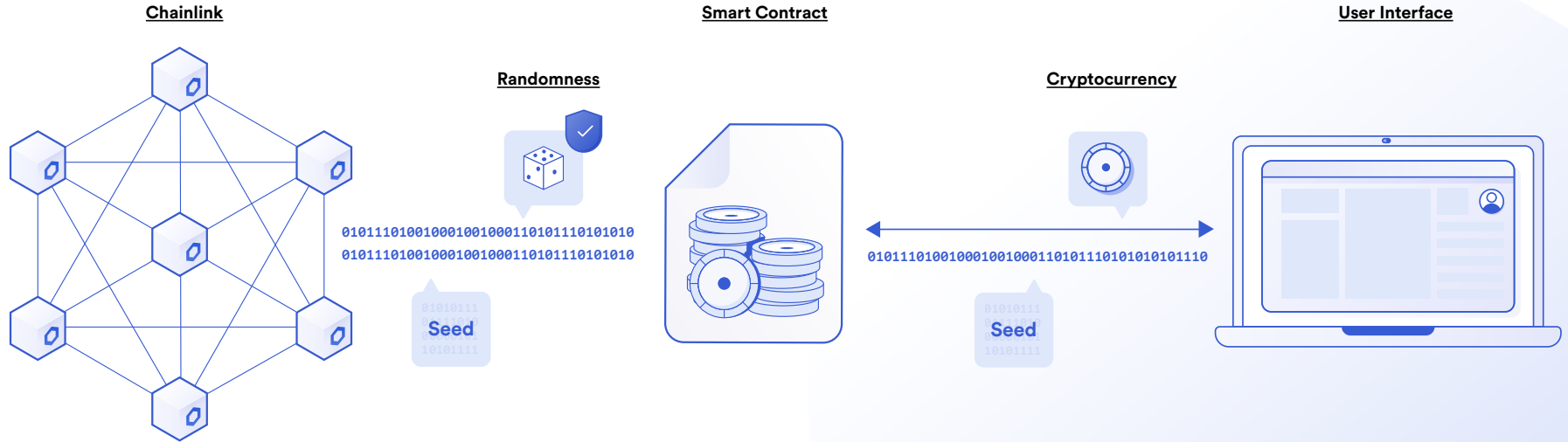
Chainlink VRF Provides Verifiable Randomness



Chainlink VRF Can Use Multiple Seeds



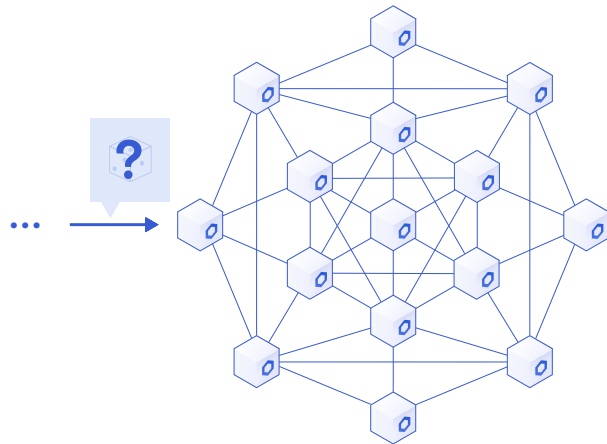
Chainlink VRF Gains Availability with More Nodes



Threshold VRF

Off-Chain

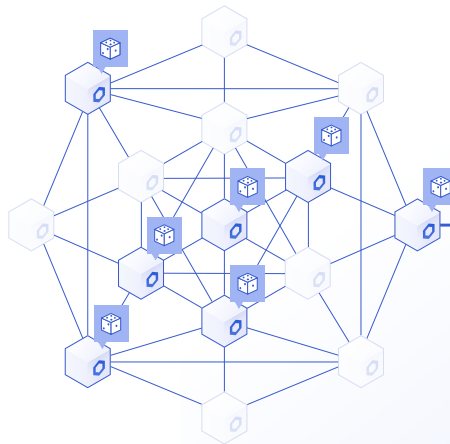
Chainlink Nodes



A new request for data is sent to the Chainlink network.

Off-Chain

Chainlink Nodes

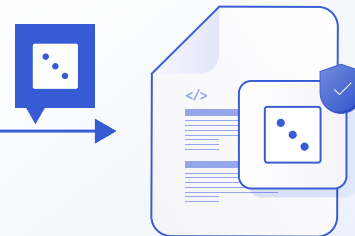


Chainlink nodes respond and aggregate responses with a required number of participants (e.g. 7).

Once the threshold is met, the result is broadcasted in a single transaction.

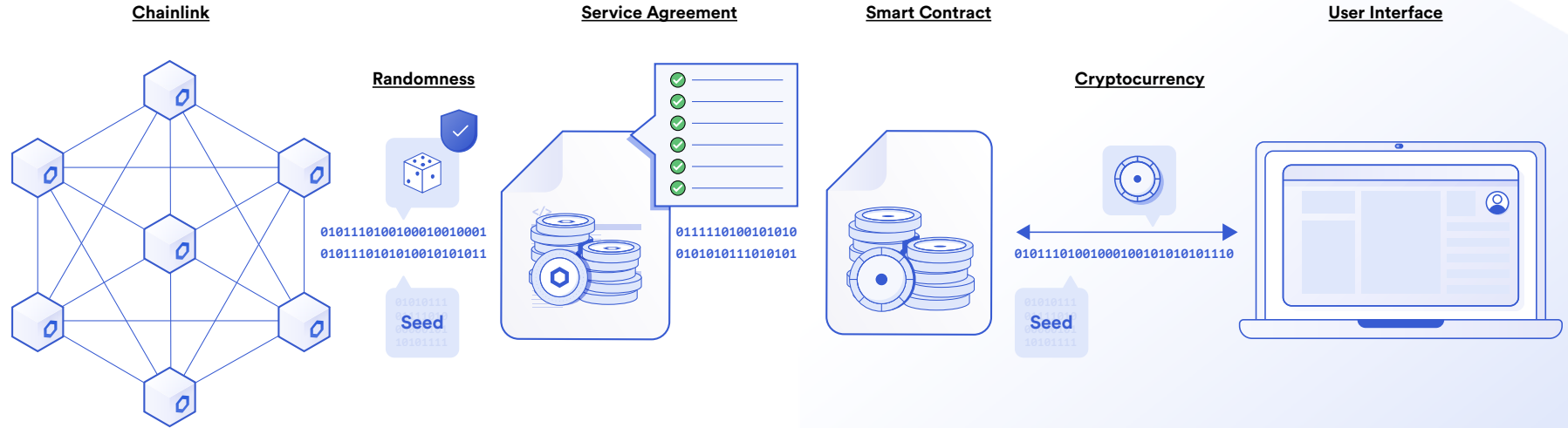
On-Chain

Smart Contract



The result is verified, updated and made available for smart contract applications.

Chainlink VRF Can Utilize Cryptoeconomic Security



Chainlink VRF: On-chain Verifiable Randomness

```
</>  
  
function requestRandomness(bytes32 _keyHash,  
uint256 _fee, uint256 _seed) public returns  
(bytes32 requestId);  
  
function fulfillRandomness(bytes32 requestId,  
uint256 randomness) external {  
    // Do something with randomness  
}
```

**Available
On Testnet
Today**

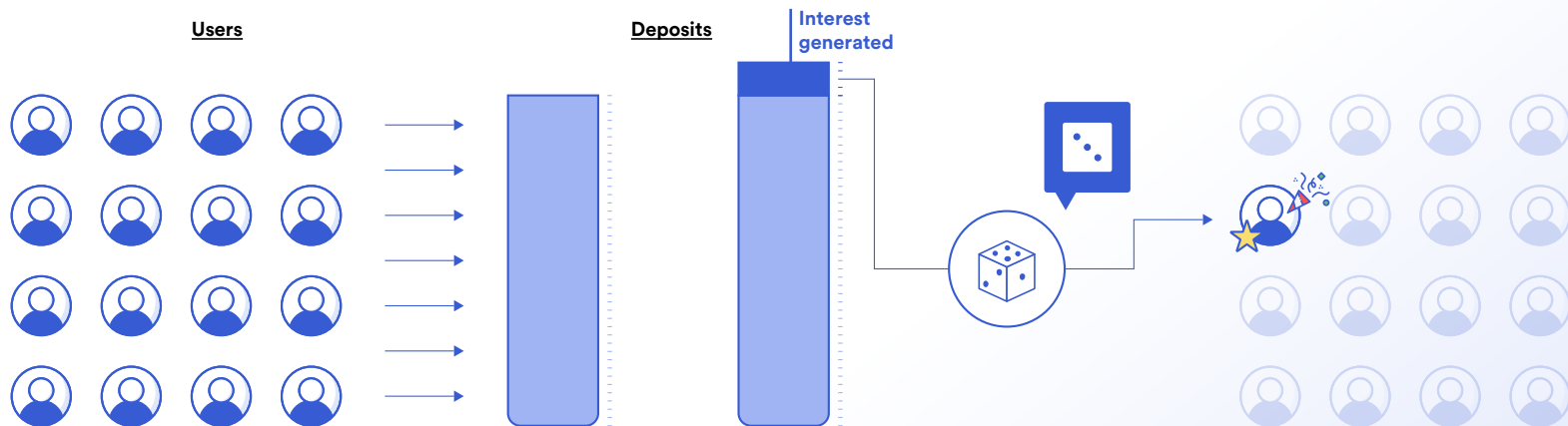
<https://docs.chain.link/docs/chainlink-vrf>



Chainlink

Pool Together

The no-loss savings game

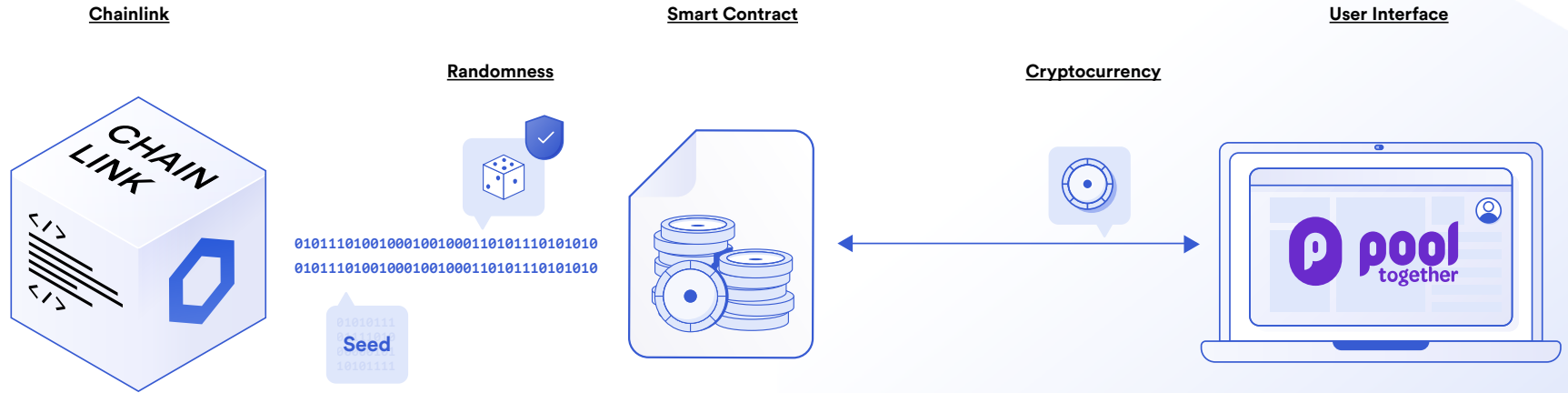


Users deposit into PoolTogether.

PoolTogether uses DeFi lending protocols to generate interest on deposits.

PoolTogether uses randomness to select a winner, who gets all interest accrued that week. Underlying deposits still belong to each user.

Chainlink VRF Providers Verifiable Randomness

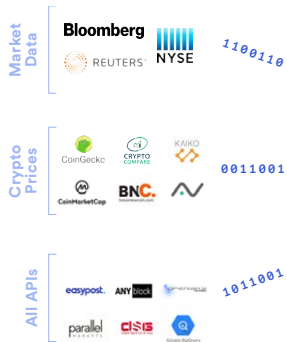


Connecting Smart Contracts to All Inputs and Outputs

Smart Contracts
& Blockchains



Contract Inputs



Contract Outputs



Join Our Team



Build great open source software that enables the next generation of DeFi and many other smart contract types.

We're an idea meritocracy where the best ideas win.

We're a remote team working with great people all over the world.

careers.chain.link

Thank You

Disclaimer: This presentation is for informational purposes only and contains statements about the future, including anticipated programs and features, developments, and timelines for the rollout of these programs and features. These statements are only predictions and reflect current beliefs and expectations with respect to future events; they are based on assumptions and are subject to risk, uncertainties, and change at any time. There can be no guarantee that any of the contemplated programs or features will be implemented as specified nor any assurance that actual results will not differ materially from those expressed in these statements, although we believe them to be based on reasonable assumptions. All statements are valid only as of the date first presented. The statements in this presentation also may not reflect future developments due to user feedback or later events and we may not update this presentation in response.