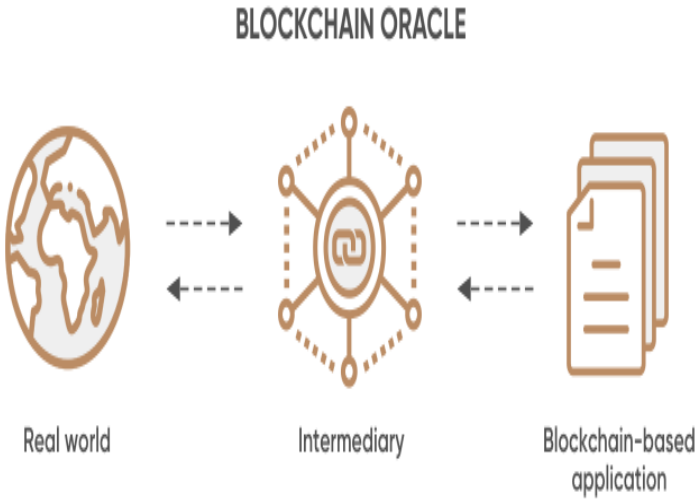
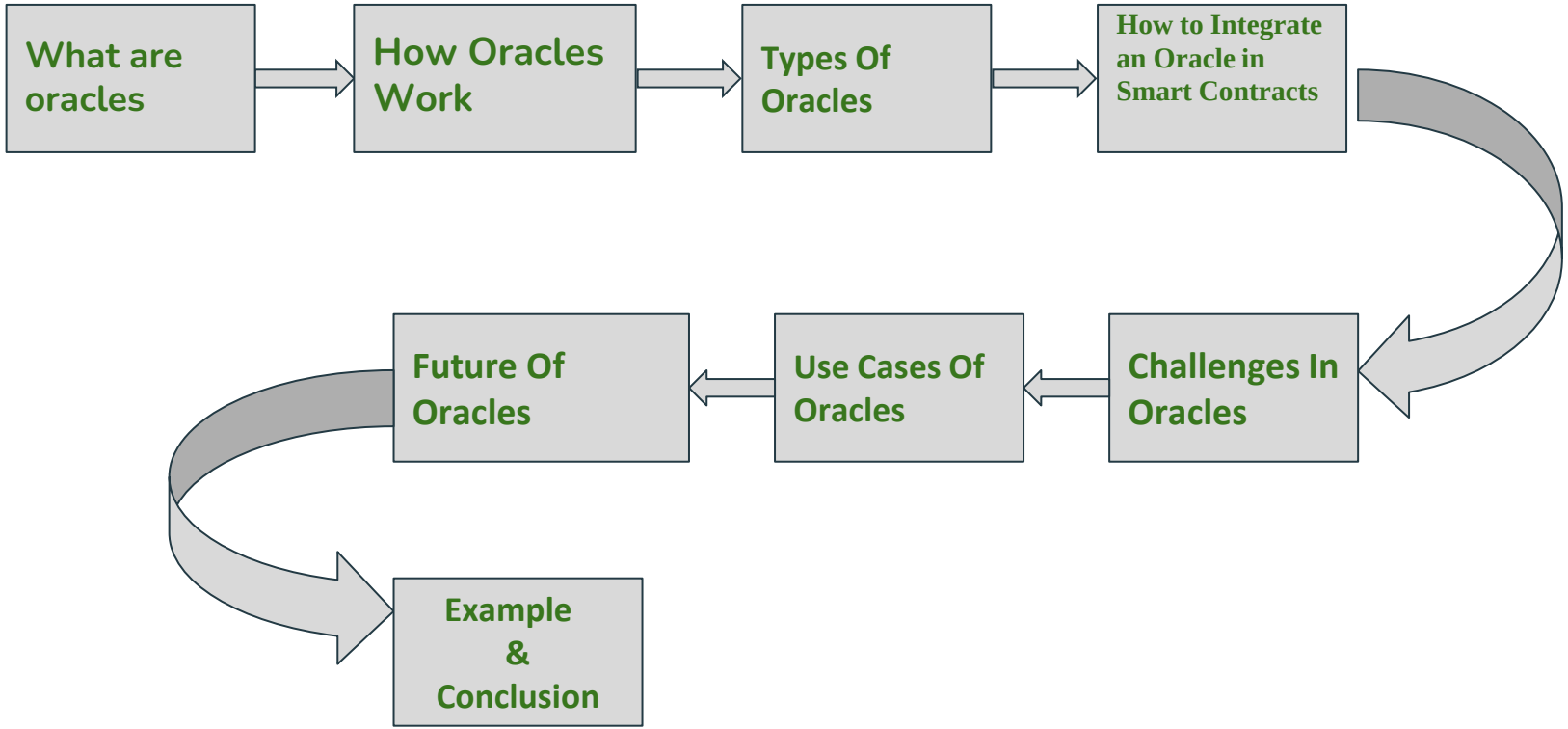
ORACLES

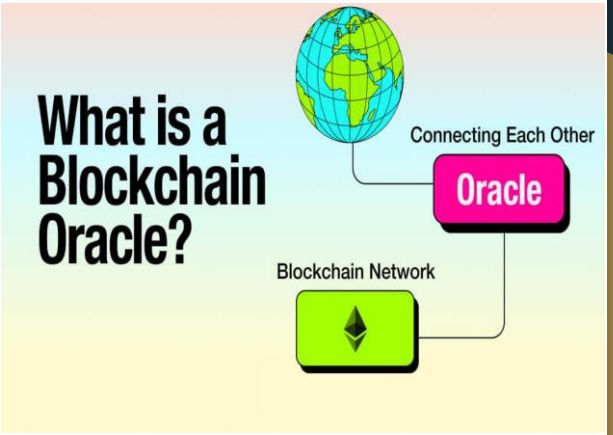

BLOCKCHAIN ORACLE
Real world
Intermediary
Blockchain-based application

Topics Covered In Presentation



Oracle?

- **Definition**: An oracle is an interface that delivers external, real-world data to smart contracts.
- **Purpose**: It bridges the gap between the isolated blockchain environment and the external world.
- **Data Examples**: Stock prices, weather reports, IoT device data, flight delays, etc.



Data Providers

Oracles are entities. They supply external data to blockchains. This data can be real-world events. It could also be price feeds or weather information.

Bridge to Reality

Blockchains are isolated systems. Oracles act as bridges. They connect them to the outside world. They enable smart contracts to interact with external data.

Essential for Smart Contracts

Many smart contracts require external data. Oracles make this data available. This allows contracts to execute based on real-world conditions.

How Oracles Work

1. **Data Flow**: Oracles fetch external data and send it to smart contracts.

1. **Push vs. Pull**:

   Push: Oracle sends data to smart contract.

   Pull: Smart contract requests data from the oracle.

1. **Trust**: Oracles ensure data is authentic and can be signed digitally to prove its source.

Example Case: Weather-Based Insurance Smart Contract

1. **Scenario**: A farmer takes out an insurance policy that pays out if it doesn't rain for a certain number of days.
2. **Push**: An oracle continuously monitors weather data. If the conditions (no rain) are met, the oracle sends the data directly to the smart contract, triggering the payout.
3. **Pull**: The smart contract, at a specified time, requests weather data from the oracle to check if the payout conditions are met.
4. **Trust**: The oracle provides the weather data and signs it digitally, proving it came from a trusted source.

How to Integrate an Oracle in Smart Contracts

Step 1: Deploy a smart contract that needs external data.

Step 2: Connect to an Oracle service (e.g., Chainlink).

Step 3: Set up a request-response mechanism.

Step 4: Implement data validation to prevent manipulation.

```
// In your contract:

function requestPriceData() public {

    bytes32 queryId = oracle.requestData("latest ETH price", address(this),

}


function fulfillPriceData(bytes32 _requestId, bytes memory _data) public {

    // Parse and validate _data

    // Update contract state with validated data

}
```

Designing contract program:

```
pragma solidity ^0.8.0;


interface ChainlinkOracle {

    function requestData(bytes32 _queryId, address _callbackContract, bytes4

    function fulfillData(bytes32 _requestId, bytes memory _data) external;

}


contract MyContract {

    ChainlinkOracle private oracle;


    function requestPriceData() public {

        bytes32 queryId = oracle.requestData("latest ETH price", address(thi

    }


    function fulfillPriceData(bytes32 _requestId, bytes memory _data) public

        // Parse and validate received price data

        // Update contract state with validated data

    }

}
```

Types of Oracles: On-Chain and Off-Chain

On-Chain Oracles

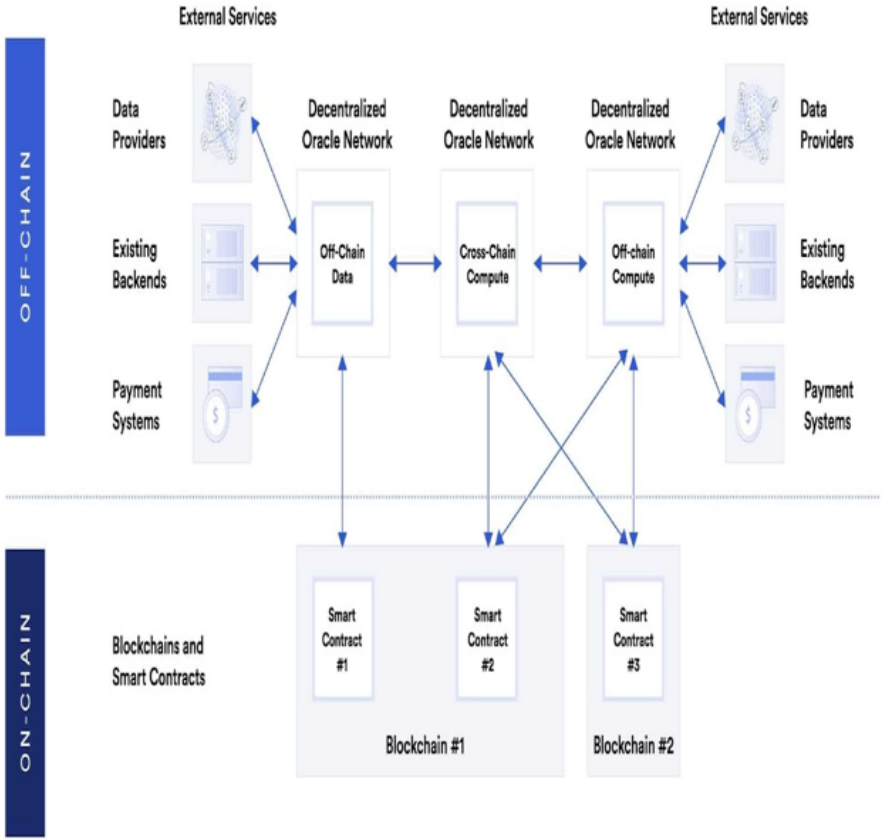
On-chain oracles are smart contracts. They reside directly on the blockchain. They retrieve data from within the blockchain network.

Off-Chain Oracles

Off-chain oracles collect data from external sources. They then transmit it to the blockchain. These are external entities.

Centralized Oracles

Centralized oracles are controlled by a single entity. They are a sing source of truth for data. However, they can be a point of failure.



Challenges in Oracle Design and Implementation

Security Risks

Oracles can be vulnerable to attacks. Data manipulation is a major concern. This compromises smart contract integrity.

Data Accuracy

Ensuring data accuracy is critical. Oracles must provide reliable information. This avoids incorrect contract execution.

Centralization Concerns

Centralized oracles create single points of failure. This undermines blockchain's decentralized nature. Decentralization enhances trust.

Solutions to Oracle Problems

Decentralized Oracles – Use multiple sources to validate data.

Reputation Systems – Rank Oracles based on accuracy history.

Cryptographic Proofs – Mechanisms like TLSNotary verify data integrity.

Economic Incentives – Oracles stake tokens and get penalized for incorrect data.

Use Cases for Oracles in Blockchain Applications

1

Supply Chain Management

Oracles track goods. They verify product origin. This ensures transparency. It improves supply chain efficiency.

2

Decentralized Finance (DeFi)

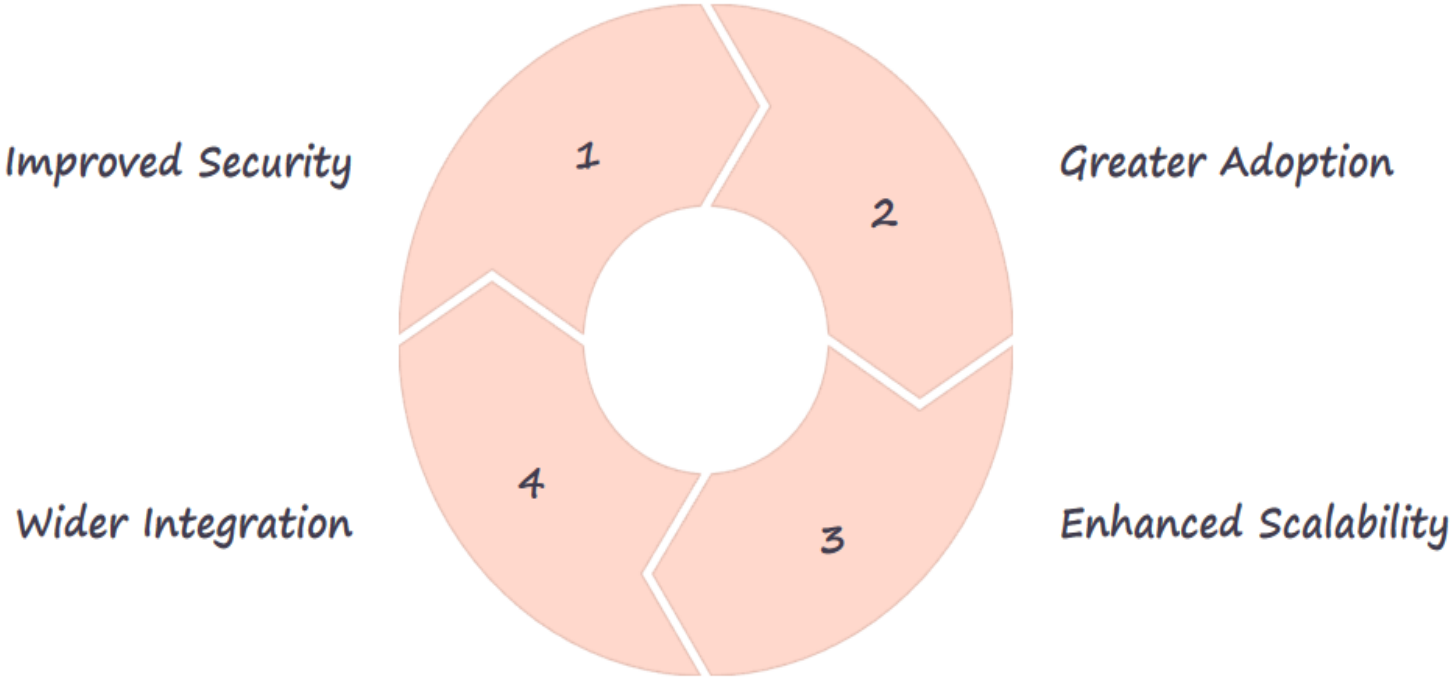
Oracles provide price feeds. These are used in lending. They're also used in trading platforms. They enable accurate financial transactions.

3

Insurance

Oracles verify event occurrences. This automates insurance payouts. It increases efficiency. It also reduces fraud.

The Future of Oracles in the Blockchain Ecosystem



Example For Oracles In Block Chain

1. A **farmer** buys an insurance policy stored as a **smart contract** on the blockchain.
2. The smart contract needs **real-time weather data** to decide if a payout is necessary.
3. A **blockchain Oracle** fetches weather data from an **external weather API**.
4. If rainfall is **below 10mm for a week**, the Oracle sends this data to the **smart contract**.
5. The smart contract **verifies the data** and automatically **releases the payout** to the farmer.
6. This process ensures **transparency, automation, and trust** without manual claims.
7. The Oracle acts as a **bridge** between real-world data and the blockchain, enabling **smart contracts to function autonomously**.

Conclusion:

Oracles play a crucial role in bridging **blockchain smart contracts** with real-world data. They enable **automation, trust, and accuracy** by securely fetching external information like **weather, stock prices, or IoT data**. Decentralized Oracles enhance **security and prevent data manipulation**, ensuring **reliable execution** of smart contracts. However, trust issues in centralized Oracles highlight the need for **verification mechanisms**. With Oracles, blockchain technology can be integrated into **finance, insurance, supply chains, and beyond**. Their continued development will drive **real-world blockchain adoption**.


Thank you