ORACLES

Blockchain technologies, **lecture** 7

Course overview

- Oracles
  - Blockchain middleware, general concepts
  - Oracle's problem: decentralization and security
- Oracles design patterns.
- Case study: Chainlink architecture.

ORACLES

ORACLES access off-chain data

- **Third party services that provide smart contracts with external information.**
- Broaden the scope in which smart contracts operate.
- Oracles classification:
  - Source: hardware, software or human
    - Software Oracles (deterministic oracles) transmit information from online databases, servers, websites etc.
    - Hardware Oracles: translates real-world events in information that can be understood by smart contracts; sensors; Supply Chain applications
    - Human oracles, cryptographically identified, transmit respond to arbitrary inquires.
  - Direction of information: inbound or outbound
  - Trust: centralized or decentralized

ORACLES provide data and off-chain computations

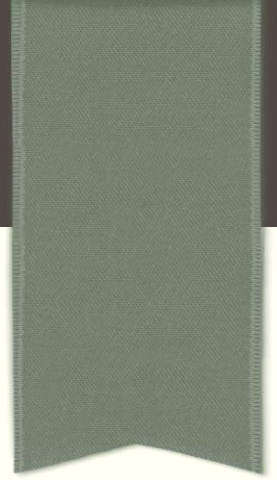
- **Data feeds (existing data sources or computations) that connect blockchain to off-chain information.**
- Inbound or outbound
  - **Inbound oracles** transmit information from external sources to smart contracts
    - Temperature measured by a sensor
  - **Outbound oracles** send information from smart contracts to the external world.
    - Smart lock: outbound oracle receives information from smart contract to unlock the smart lock.
  - **Cross chain oracles** read and write information between different blockchains.
  - **Compute-enabled oracles** RNG

ORACLES provide data and off-chain computations

- **Data feeds that connect blockchain to off-chain information.**
- Applications:
  - payments;
  - exchange rates, decentralized finance (DeFi) applications;
  - predictions, bets;
  - weather reports (insurance calculations);
  - Output Oracles: IoT sensors for supply chain;
  - Dynamic NFTs, gaming, **verifiable randomness** (fairly select a winner in a lottery);
  - Insurance smart contracts etc.
  - Gaming.

ORACLES provide data and off-chain computations

- **Data feeds that connect blockchain to off-chain information.**
- **Scale blockchain:** off-chain computations.
- Functionalities:
  - Receives requests from smart contracts;
  - Send data to external systems;
  - Extract data;
  - Compute: verifiable RNG, aggregate data etc.;
  - Format data in blockchain compatible formats;
  - Validate: zero-knowledge proofs, Trusted Execution Environment (TEE) attestations, TLS signatures etc.



ORACLE PROBLEM

ORACLES access off-chain data

- **Data feeds that connect blockchain to off-chain information.**
- Blockchain: deterministic transactions, same result on every node, independent of the moment in time the transaction is processed.
- API calls may result in different results and in the impossibility to reach consensus.

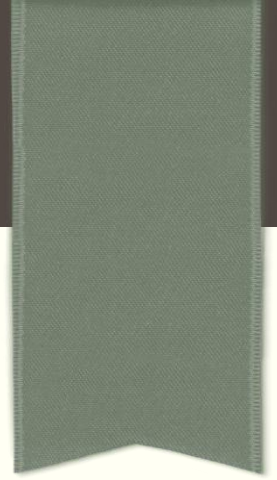
- An oracle records the result of calling an external API through a blockchain transaction (blockchain middleware).
- Typically, an oracle is a smart contract accessing some of-chain components that can query APIs.

ORACLES centralized/decentralized

- Smart-contracts cannot directly interact with data existing outside blockchain environment.
- Centralized oracle: single point of failure.
- Security risks (who controls the API?).
- Blockchain immutability: faulty data cannot be reversed.
- Solutions:
  - **Decentralized Oracle (DON – decentralized oracle network) combines multiple data sources and multiple independent oracles.**
  - prove data validity

ORACLES centralized/decentralized

- Solutions:
  - Decentralized Oracle (DON – decentralized oracle network) combines multiple data sources and multiple independent oracles.
    - Requires predefined standard data format.
    - Inefficient, gathering answers from multiple sources requires time.
    - Data providers may require fees.
  - prove data validity:
    - No need to trust the oracle
    - Data providers don’t have to modify services in order to become compatible with Blockchain technologies.
  - Decentralized oracles distribute trust between many participants.



ORACLES DESIGN PATTERNS

ORACLES centralized/decentralized

- Key functionalities:
  - Collect data from off-chain source
  - Transfer the data on-chain with a signed message.
  - Store the data in a smart contract's storage.

  - Once the data is available in a smart contract's storage, it can be accessed via message calls.

  - Immediate-Read oracles:
    - Examples: data about persons/organizations, data issued by organizations.

ORACLE design patterns

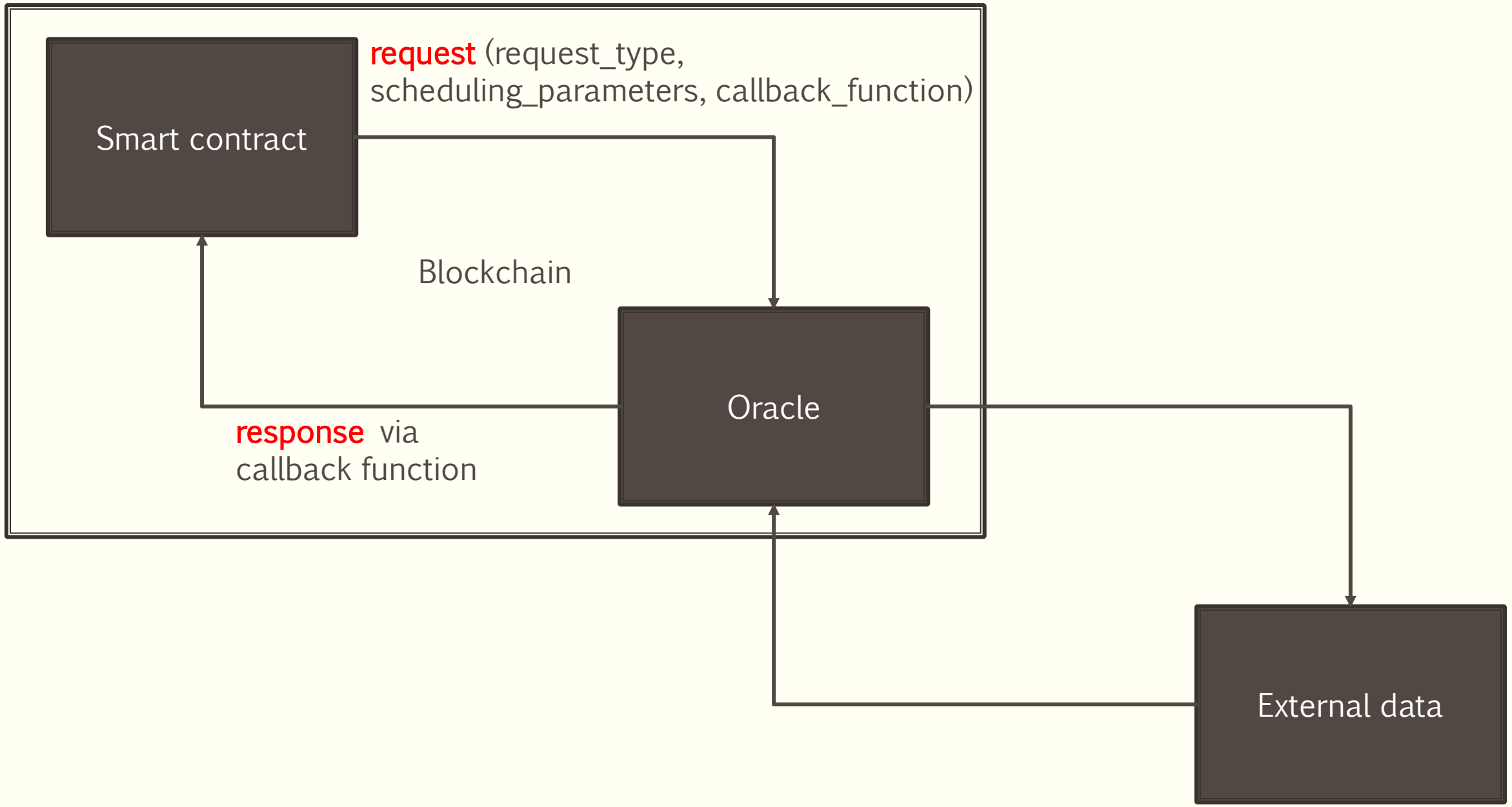
- Publish-subscribe pattern.
- Data is expected to change frequently: prices, weather, traffic data etc.
- Similar to RSS feed.
- Subscribers listen to oracle contracts updates.
- In Ethereum: event logs, can be considered “push” service.
- Publishers categorize messages into classes. Subscribers may express an interest in one or more classes.

ORACLE design patterns

- Request-response pattern.
- Data can’t be stored in a smart contract.
- On-chain smart contract and off-chain infrastructure.
- Requests include scheduling parameters and callback functions.
- The Oracle may require payment for processing the request.

ORACLE design patterns

- Request-response pattern.
- Oracles receives a query from a Dapp with arguments detailing the data requested, callback functions and scheduling parameters.
- Parse the query
- Check payment
- Retrieve data from off-chain source and encrypt it if necessary.
- Broadcast the transaction to the network.
- Schedule any further necessary transactions.


Smart contract
request (request_type,
scheduling_parameters, callback_function)
Blockchain
Oracle
response via
callback function
External data

CHAINLINK

Chainlink

- USER-SC Requesting contract.
- CHAINLINK-SC On chain contract responding to USER-SC requests.
- Adapter: interface with CHANLINK-SC on-chain smart and workloads across external services.

Main components:

- Reputation contract:
- Order-matching contract
- Aggregating contract

Chainlink

- **Reputation contract**: keeps record of the service provider performance metrics.
  - Incentivize and penalize reporting contracts
  - Historical performance history.
- **Order-matching contract:** delivers the request to Chain-link nodes and take their bids on the request and select the number and the types of nodes to fulfill request
- **Aggregating contract**: calculates weighted answer. The validity of each response is reported to the reputation contract.

- Requesting contract pay with LINK built in accordance with ERC-20 standard.
  - Subscription
  - Direct funding

Chainlink applications VRF, PRICE FEEDS

- Chainlink price feeds:
  - Price data aggregators (data source - decentralization)
  - Node operators (individual node operators - decentralization)
  - Oracle network aggregation.

- Verifiable random number generator (RNG).
  - Random values with on-chain cryptographic proof.
  - Applications: Gaming, Security, DeFi, advertising.
  - A Chainlink node generates randomness.
  - After generating the randomness, it will send the cryptographic proof to the VRF contract. **Verifiable Random Function**: a provably fair and verifiable random number generator (RNG).
  - After being approved the randomness is publicly displayed on the blockchain.

faucets.chain.link

Your wallet is connected to Ethereum Sepolia, so you are requesting Ethereum Sepolia LINK/ETH.
If you need testnet tokens for a different network, switch the network in your wallet

Wallet address

0x50497e8c0272d6c3d37fce08f8c3

Request type

25 test LINK

Verify request

Success!

Send request

Need more testnet ETH? Get

Enter wallet address — Initiating — Waiting for confirmation — 4 Tokens transferred

Waiting for confirmation

Transactions have been initiated. Waiting for confirmation.

Request type	Transaction hash
25 test LINK	0xce2021315bd4870753898f5f2d8b6480ee690f9fbe495aba75d8304a4f7e9ec9

Close

vrf.chain.link/goerli/new

Chainlink | VRF

NEW Introducing Transporter, a new bridging app powered by CCIP. Go c

Home /

Create Subscription

Create a Subscription ID using your wallet address. The Subscription ID will be used in your contract when requesting a random value. Learn more

Please connect your wallet

To create a Subscription please connect to your wallet first.

Connect wallet

Need help or have questions? Talk to an expert or visit the VRF webpage

Subscription

Actions

Status	ID	Admin	Consumers	Fulfillments	Balance
Active	10961	0x5049...4c10	0	0	5 LINK

No consumers

Your subscription is ready. You can now add consumers.

Add consumer