Chapter 11: Oracles.

Oracles are are systems that can provide external data sources to Ethereum smart contracts.

Ideally oracles are systems that are trustless, meaning that they do not need to be trusted because they operate on decentralized principles.

Due to the fact that sometimes, miners can be evil and only include their blocks for which their source of randomness would win, it is safer to introduce an extrinsic source of randomness to the blockchain. This is what Oracles seek to provide.

Oracles, ideally, provide a trustless (or at least near-trustless) way of getting extrinsic (i.e., “real-world” or off-chain) information, such as the results of football games, the price of gold, or truly random numbers, onto the Ethereum platform for smart contracts to use.

Oracles can therefore be thought of as a mechanism for bridging the gap between the off-chain world and smart contracts.

An Oracle can feed in any external data to the blockchain smart contract.

All oracles provide a few key functions, by definition. These include the
ability to:
Collect data from an off-chain source.
Transfer the data on-chain with a signed message.
Make the data available by putting it in a smart contract’s storage.

Once the data is available in a smart contract’s storage, it can be accessed by other smart contracts via message calls that invoke a “retrieve” function of the oracle’s smart contract; it can also be accessed by Ethereum nodes or network-enabled clients directly by “looking into” the oracle’s storage.

The three main ways to set up an oracle can be categorized as
Request–response, (Hard)
Publish-subscribe, (Medium) and

Immediate-read. (Simplest)

Immediate-read:
When you need data for quick decisions, and you might not need them again.

Publish-subscribe:
where an oracle that effectively provides a broadcast service for data that is expected to change (perhaps both regularly and frequently) is either polled by a smart contract on-chain, or watched by an off-chain daemon for updates.

Interested parties must either poll the oracle to check whether the latest information has changed, or listen for updates to oracle contracts and act when they occur.

Request–response:
The request–response category is the most complicated: this is where the data space is too huge to be stored in a smart contract and users are expected only to need a small part of the overall dataset at a time.

Two common approaches to data authentication are:
Authenticity Proofs and
Trusted Execution Environments (TEEs).

Authenticity proofs are cryptographic guarantees that data has not been tampered with.

Town Crier is an authenticated data feed oracle system based on the TEE approach; such methods utilize hardware-based secure enclaves to ensure data integrity.

However, oracles can also be used to perform arbitrary computation, a function that can be especially useful given Ethereum’s inherent block gas limit and comparatively expensive computation costs.

ChainLink has proposed a decentralized oracle network consisting of three key smart contracts —
a reputation contract,
an order-matching contract, and
an aggregation contract —
and an off-chain registry of data providers. The reputation contract is used to keep track

of data providers’ performance.

As you can see, oracles provide a crucial service to smart contracts: they bring external facts to contract execution. With that, of course, oracles also introduce a significant risk — if they are trusted sources and can be compromised, they can result in compromised execution of the smart contracts they feed.

Before working with an oracle, first, consider the Trust Model. Don't assume, set facts straight, else, you may be undermining the security of your smart contracts.