Creating Smart Contracts with Solidity

Ethereum networks, wallets and development tools

Ethereum networks, software and development tools

What is Ethereum

- Ethereum is a network, made up of many communities, and a set of tools which enable people to transact and communicate without being controlled by a central authority*
- Ethereum is also a technology and a protocol for running such networks

Ethereum Networks

- Ethereum Mainnet
- Ethereum Testnets
- Private/Development networks

Ethereum Mainnet

- Mainnet is the primary public Ethereum production blockchain, where actual-value transactions occur on the distributed ledger.
- This is the network that underlines the ETH currency

Ethereum Testnets

- These are networks used by protocol developers or smart contract developers to test both protocol upgrades as well as potential smart contracts in a production-like environment before deployment to Mainnet
- Sepolia
  - Mostly for app testing
- Goerli
  - Mostly for testing protocol upgrades/staking

Private/Development networks

- Mostly used for local development and special purposes
- Can be run as a single node locally or distributed over many nodes
- Can be set with different consensus algorithms Proof of Work, Proof of Stake, Proof of Authority

Nodes & Clients

- A "node" is any instance of Ethereum client software that is connected to other computers also running Ethereum software
- The Ethereum network has two types of clients (layers)
  - The execution client listens to new transactions broadcasted in the network, executes them in EVM, and holds the latest state and database of all current Ethereum data.
  - The consensus client implements the proof-of-stake consensus algorithm, which enables the network to achieve agreement based on validated data from the execution client.

Nodes & Clients

- Most widely used execution clients :
  - Geth - Golang
  - Erigon - Golang
  - Besu - Java
  - Nethermind - .NET

Nodes & Clients

- Most widely used consensus clients
  - Lighthouse - Rust
  - Lodestar - Typescript
  - Nimbus - Nim
  - Teku – Java
  - Prysm - Golang

Ethereum Accounts & Wallets

- Definitions
  - An Ethereum account is an entity that can send transactions and has a balance.
  - An Ethereum account has an Ethereum address, like an inbox has an email address. You can use this to send funds to an account.
  - A wallet is a product that lets you manage your Ethereum account. It allows you to view your account balance, send transactions, and more.

Ethereum Accounts & Wallets

- Types of wallets
  - Hardware/Physical wallets
  - Mobile applications
  - Browser wallets
  - Browser extensions
  - Desktop applications

WalletConnect

- WalletConnect is the Web3 messaging layer and a standard to connect blockchain wallets to dapps.
- Interoperatbility Standard for both
  - dapp builders/creators
  - Wallet creators/maintainers

WalletConnect supported wallets

- MetaMask
- OneKey
- Web3Auth
- Also several non-FOSS options

Smart Contracts

- A "smart contract" is simply a program that runs on the Ethereum blockchain. It's a collection of code (functions) and data (state) that resides at a specific address on the Ethereum blockchain.
- They are also and Ethereum account, so they can hold balance and execute transactions
- Smart contracts cannot be deleted by default, and interactions with them are irreversible.

Smart Contracts langauges

- Solidity
  - OOP/C++ like. Oldest and most widely used
  - De facto standard for writing Smart Contract
- Vyper
  - Python like with string typing.
  - Simpler , has intentionally less features than Solidity for easier audit

Dapps

- Dapp is application where the backend uses decentralized services
- Smart contracts usually underline the core business logic of a dapp
- IPFS is often used for storage of files

Dapps pros

- Zero Downtime
  - The whole network is serving the SC, so it’s much harder do DoS an app
- Privacy
  - Anonymous interactions by default
- Resistance to censorship
  - No single entity has authority over the transactions
- Complete data integrity
  - Blockchain guarantees immutability and verifiability of the data
- Trustless computation/verifiable behavior
  - Anybody can analyse a deployed SC , no need to trust central authority

Dapps cons

- Harder to maintain
  - Data is immutable, so “data migrations” are harder and require more effort/thought
- Slow / Performance overhead
  - Scaling is hard, because of the need for ensuring consistency, security and transparency. Consensus also takes time
- Network congestion
  - Limited number of transactions per second. Currently at around 10-15 tx/s
- Harder to create smooth user experience
  - Often require additional local software setup before use
- Centralization
  - Pursuit of better UX can often lead to centralization of the service

Exercise Time:
Building a private network with PoA consensus