Block Chain, Bitcoin & Security

…But truly Bitcoin, Proof of Work and Smart Contracts Development

EPITA - 2021

Part 3: Ethereum

1. Introduction
2. How does it work?
3. Ethereum ecosystem

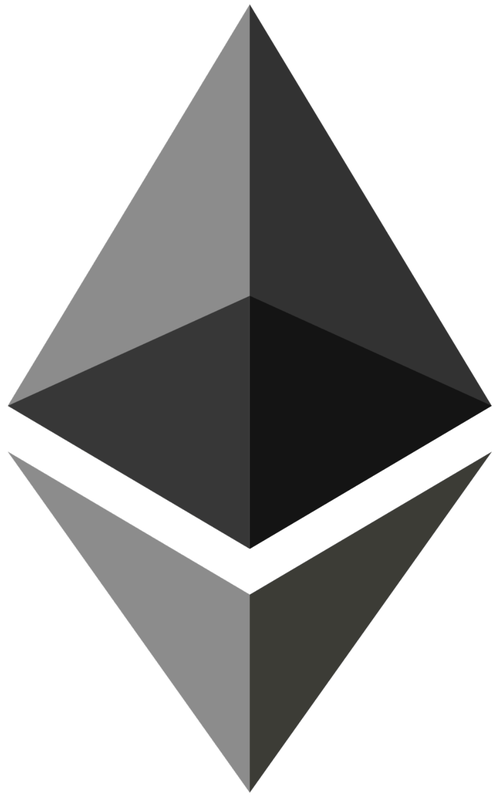
Part 3: Ethereum

1. Introduction
2. How does it work?
3. Ethereum ecosystem

Introduction

Why does Ethereum exist?

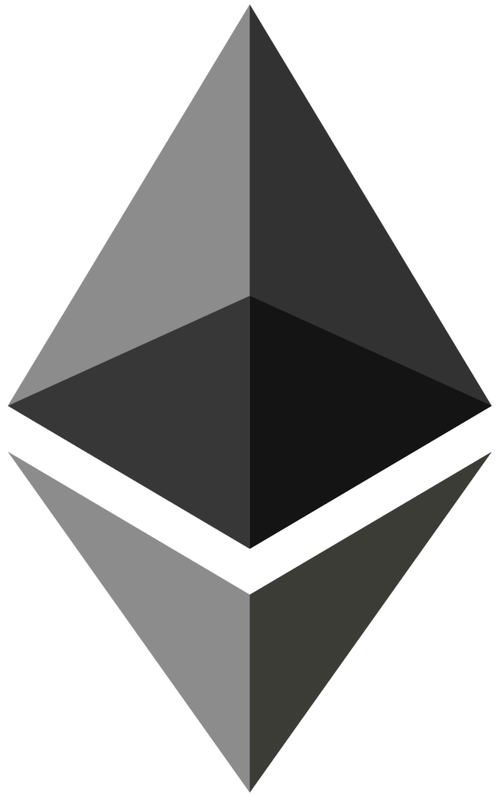
- Bitcoin was “just” a decentralised currency
- There was the need to decentralise more things (real estate ownership, artwork ownership, games, social networks…)
- Having to build a new network with enough peers and hash power for each of these was a solution, but a tough work



Introduction

What does it do?

- Ethereum is a platform to upload Decentralised Apps (Dapps), based on what are called smart contracts
- Just like Bitcoin, Ethereum takes profit of a decentralised network, but anyone can code programs called “smart contracts” on it
- It uses “ether” as its currency to reward miners and to pay transaction fees



Ethereum

Goals

- The goal of Ethereum is to decentralise the Internet
- It would do it by providing a secure, decentralised internet
- It truly enables “cutting the middle-man”: Uber, Amazon, eBay, YouTube (music producing)… And fund raising (Kickstarter => ICOs)

Part 3: Ethereum

1. Introduction
2. **How does it work?**
3. Ethereum ecosystem

How does it work?

A few reminders

- Ethereum is still based on a Blockchain, so most things which are true for Bitcoin are true for Ethereum: Block hashes, Proof of Work (for now), P2P Network
- You also still have wallets (private key, public key), transactions, mining rewards…

How does it work?

A few things about smart contracts

- The real innovation here are the smart contracts: these are simple byte codes that are ***immutable*** and ***won't be owned or controlled by any user*** (unless…)
- Those smart contracts will have a public address and their byte code will be made public, so that the community can audit its features and security
- Smart contracts functions can be called by anyone, but only accept external calls…

How does it work?

What “external calls only” entail

- You NEVER need to log in to use an Ethereum smart contract
- CRON jobs cannot be done in Ethereum (unless…)
- It may be vulnerable to DDoS attacks
- To counteract that, Ethereum includes a “transaction fee” for (almost) every function call
- In consequence, intensive computing tasks can cost a lot… And are forbidden by Ethereum
- It is very hard for an Ethereum smart contract to use data from “the outside world”

How does it work?

The “unless” answers

Constraint	Solution
Smart contracts are immutable	You can reference another smart contract and use its functions
Smart contracts are not owned by anyone	You can code an “owner” in the smart contract constructor and allow him to call functions that only he can call
CRON jobs are impossible	You can code your own server to make CRON job calls from your server; services called “Oracles” also do that
Impossible to use data from the outside world	Oracles can help with that

How does it work?

Side note on “immutable”

- Immutable means it cannot change… But what cannot change?
- The hard code of any smart contract will never change. However, your smart contract may depend on state variables that you may change if you write a function dedicated to it.
- **Truly, what is immutable is the history of what has ever happened on the blockchain.**

How does it work?

Cons

- Calls to functions are much slower than by using a simple server-based API: because each function call is a transaction, the transaction must be validated and propagated through the network.
- Ethereum is a very poor database: since storing anything on Ethereum would make a copy of it on every node in the world, it is both inefficient and expensive.

Part 3: Ethereum

1. Introduction
2. How does it work?
3. **Ethereum ecosystem**

Ethereum ecosystem

Notable library and smart contracts

- ERC20: the “standard” token developed by the Open Zeppelin team. It allows the easy creation of “tokens”, that you can use to create your own currency in minutes.
- Crowd sale smart contracts: from Open Zeppelin as well, those smart contracts allowed the ICO craze of 2016-2017.
- ERC721: the “standard” non-fungible token. That means that each token may be different, cannot be divided. Ideal for any collectible.
- ERC1400: a standard for “security tokens”: these should be technically considered as ERC20, but should have additional features to allow them to be legally valuable as securities.

Ethereum ecosystem

2 languages

- Solidity: the most popular language for smart contract development. It has more library, a bigger community, open source codes are more easily audited. The language looks like JavaScript or C++.
- Vyper: another language with an emphasis on security and math functions. Some security and math features are native to the language itself.
- It is important to note that both languages will compile to give the following outputs:
  - an *abi* file (looks like a json file, has the same function as a .h in C)
  - A bytecode which is assembly for the Ethereum VM

Ethereum ecosystem

Solidity compilers and migration managers

- Truffle: develop in your favorite local IDE, unit test more easily: https://www.trufflesuite.com/docs/truffle/overview
- Hardhat: just like Truffle, easier to use and to debug: https://hardhat.org/
- Remix: develop and deploy more easily and faster through their online IDE: https://remix.ethereum.org/

Ethereum ecosystem

What will be used for this course

- Solidity: most popular programming language for Ethereum smart contracts
- OpenZeppelin-contracts: most popular library for Solidity
- Hardhat: in order to self-host smart contracts code on GitHub
- Alchemy: a hosted Ethereum node cluster to deploy smart contracts
- Metamask: a Chrome extension to sign the transactions (for deployment and function calls)
- Ropsten: a “testnet”, as opposed to “main net”, to avoid spending real ethers for tests
- MyEtherWallet: a platform that will help us ease interactions with smart contract