Smart Contracts and Blockchain Technology

Lecture 8. Smart contracts

Christian Ewerhart

University of Zurich

Fall 2022

Introduction and overview

Last lecture: Digital signatures

This lecture: Smart contracts

- What is a smart contract?
- Smart contract platforms
- Ethereum
- Solidity

Smart contracts (1)

What is a smart contract?

A **smart contract** is a piece of code that can be deployed and run on a blockchain system.

Smart contracts (2)

Characteristics of smart contracts

Immutable: Once deployed, the code of a smart contract cannot be changed.[1]

Decentralized: Execution of the code, and validation of the results, happens identically on numerous computers in parallel.

Deterministic: Starting from a given state, the code always delivers the same results (i.e., there is no randomness).

---

[1] In practice, the immutability constraint can be relaxed in a number of ways. First, smart contracts may **self-destruct**. Second, smart contracts can be embedded into **decentralized automonous organizations** that admit changes to their protocols in response to the outcome of votes. Similarly, contracts may be set up in a modular way so as to allow changes to be implemented by linking to **new modules**.

Smart contracts (3)

Some use cases

- Multi-signature wallets
- Voting and decentralized autonomous organizations (DAOs)
- Tokens, cryptoassets, and stablecoins
- Crowdfunding platforms
- Decentralized exchanges
- Smart insurance
- NFTs
- Central Bank Digital Currency (CBDC)

Smart contracts (4)

Contracts vs. smart contracts

Note: A smart contract is not a contract in the legal sense!

A **contract** in the legal meaning is an agreement between parties, creating mutual obligations that are enforceable by law.

Important characteristics of a legally enforceable contract are:

- mutual assent, expressed by a valid **offer and acceptance**
- adequate consideration and capacity
- legality

The popular phrase **"The code is the law"** reflects the anarchic mindset of early blockchain enthusiasts.

Smart contracts (5)

Contract addresses

Any Ethereum address in use is either an externally owned address or a contract account:

- An **externally owned account (EAO)** is an account created by or for human users of the Ethereum network.
- A **contract account** is an account containing code that executes whenever it receives a transaction from another account (EOA or contract).

Deployment: A transaction is sent to the contract creation address `0x0`.

Smart contracts (6)

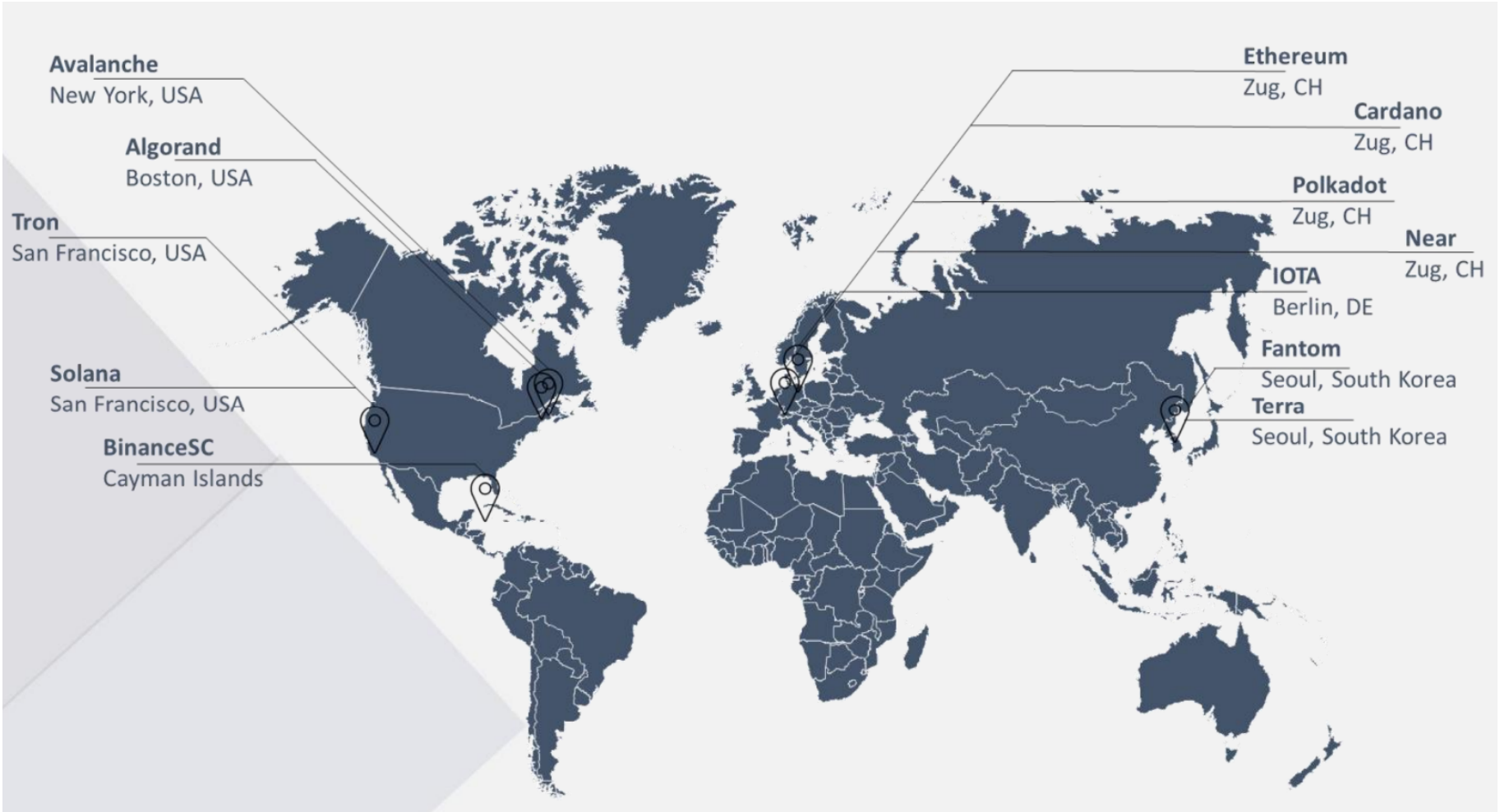
Dormant nature

Smart contracts do not run "on their own" or "in the background." Rather they stay dormant until called by a transaction.[2]

[2] The picture represents the vow scene in the movie *Corpse Bride*, a 2005 animated movie.

Smart contracts (7)

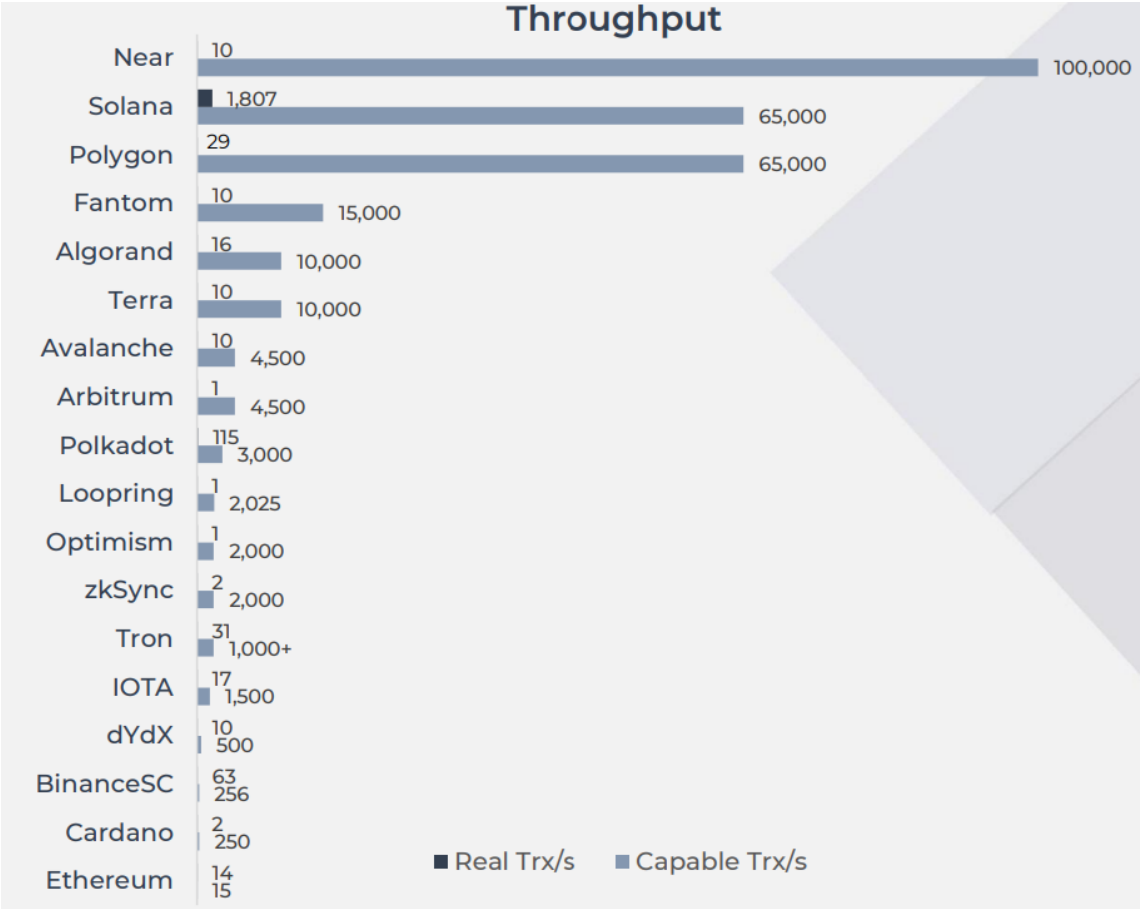
Layer-1 smart contract platforms[3]



[3] Source: Blockchain Presence

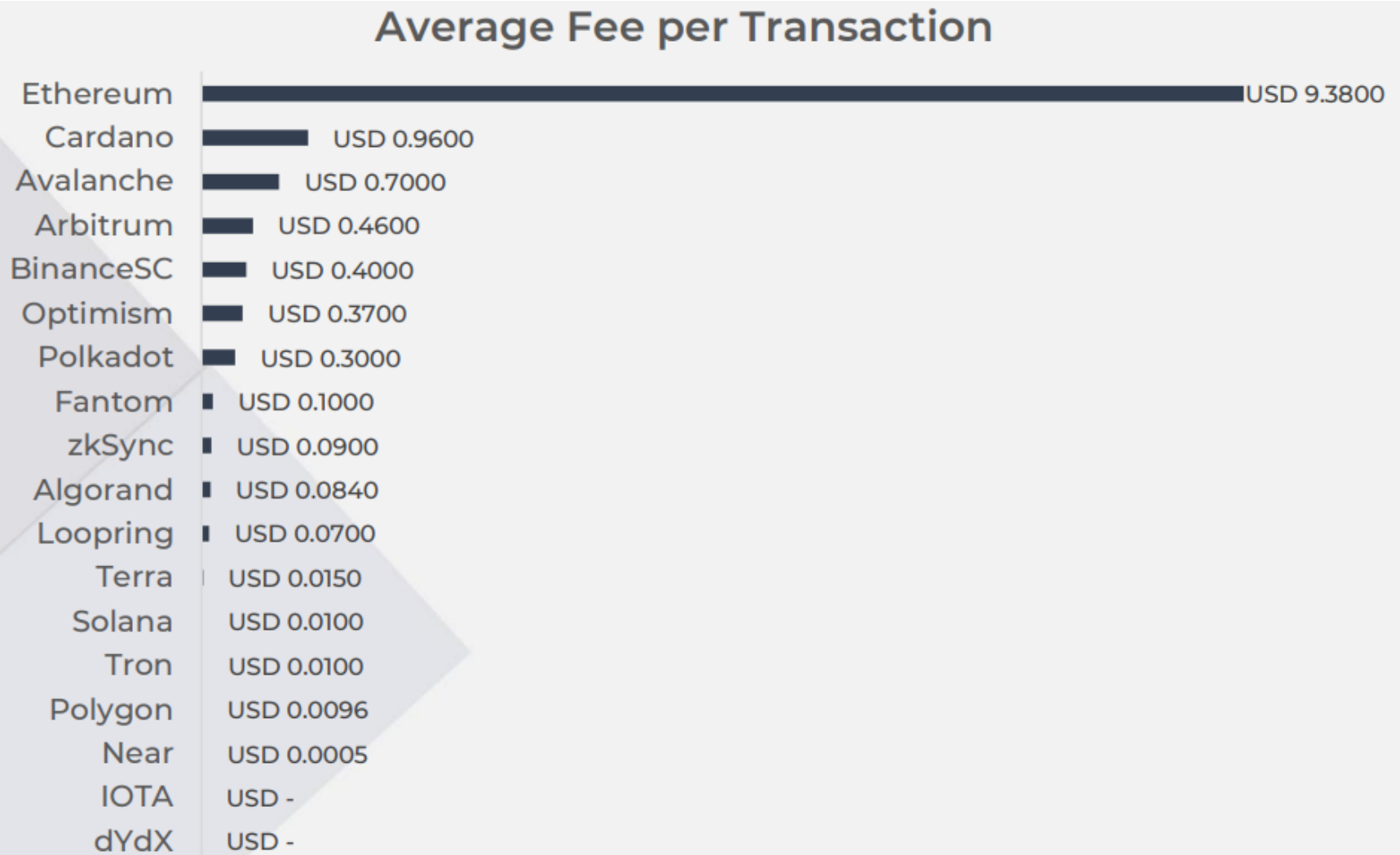
Smart contracts (8)

Transactions per second



Smart contracts (9)

Transaction costs



Smart contracts (10)

Ethereum

- the most decentralized smart contract platform
- by far the biggest developer base
- ability to be upgraded over time through EIPs
- introduced the most used token standards, e.g., ERC-20, ERC-721, ERC-1155
- merged with the Beacon chain to shift from PoW to PoS
- serves as the settlement layer for the several layer-2 solutions

Smart contracts (11)

Smart contract programming languages for Ethereum

Solidity[4]

- inspired by Javascript (the programming language that allowed the transition to an interactive web experience)

Vyper

- inspired by Python (the programming language that made coding feasible for everybody)

EVM bytecode

- the Ethereum virtual machine (EVM) is a stack-based virtual machine that executes bytecode ("machine code").[5]

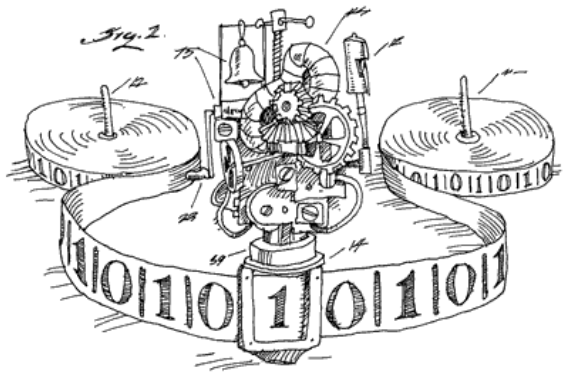
[4] See the Solidity Documentation.

[5] Virtual means that EVM instructions are not directly executed at the hardware level, but are first converted into machine code before being executed by a central processing unit (CPU). See the Ethereum yellow paper.

Smart contracts (12)

Turing completeness

A blockchain system with smart contract capability is called **Turing complete** if it can be used to simulate the operations of any Turing machine.[6]



Ethereum is Turing complete.[7]

[6] A **Turing machine**, named after the British scientist Alan Turing (1912-1954), is a theoretical model of a computer with unbounded memory executing a finite code sequence.

[7] In practice, however, gas costs render this concept rather theoretical.

Smart contracts (13)

The general structure of a smart contract written in Solidity

```
1  // SPDX-License-Identifier: GPL-3.0-only
2  pragma solidity >=0.7.0 <0.9.0;
3
4  contract Faucet {
5      // Accept any incoming amount
6      receive () external payable {}
7
8      //Give out ether to anyone who asks
9      function withdraw(uint withdraw_amount) public {
10         // Limit withdrawal amount
11         require(withdraw_amount <= 0.1 ether);
12         payable(msg.sender).transfer(withdraw_amount);
13     }
14 }
```

Smart contracts (14)

Some explanations

Comments. The double slash "`//`" introduces a comment line.[8]

License specification. It is good practice to include a license statement in your code. Under the SPDX-License `GPL-3.0-only`, others may freely copy, modify, and use your code.

Compiler instructions.[9] The line starting with "`pragma solidity`" is not part of the smart contract but indicates that the code is written in Solidity. The constraints "`>=0.7.0 <0.9.0`" specify the compiler versions with which the code is compatible.

[8] Comments are meant to inform readers of the code and have no implications for the workings of the smart contract.

[9] A **compiler** is a software tool that converts code written in a high-level programming language such as Solidity into a lower-level language such as EVM bytecode.

Smart contracts (15)

msg.sender

`msg.sender` represents the address that initiated the contract call, not necessarily the originating EOA that sent the transaction.

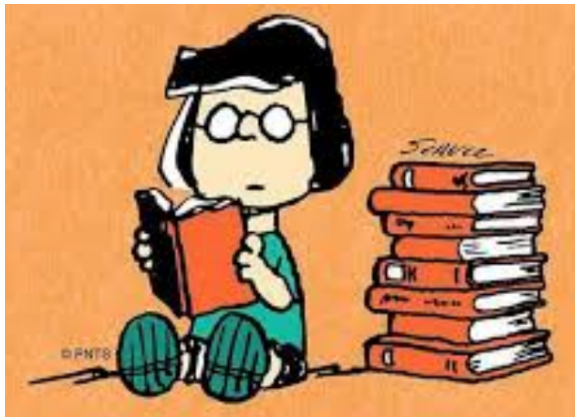
If your contract was called directly by an EOA transaction, then `msg.sender` is the address that signed the transaction, but otherwise it will be a contract address.

Smart contracts (16)

Bibliographic notes

The term "smart contracts" has been coined in the 1990s by Nick Szabo.

A useful introduction to smart contracts and Solidity can be found in Antonopoulos and Wood (2018, Chapter 7).



Smart contracts (17)

References

Antonopoulos, Andreas M., and Gavin Wood. *Mastering Ethereum: Building Smart Contracts and Dapps*. O'Reilly Media, 2018.