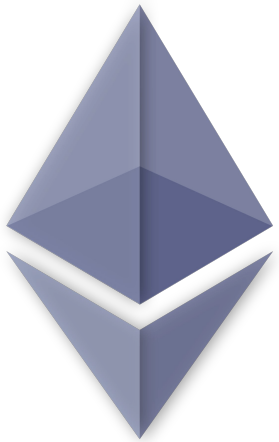
Ethereum

virtual currency,
state machines,
and programmable money



A Computer Science perspective

Who I am

Pietro De Nicolao
pd@bendingspoons.com

Software engineering lead
Algorithmic Trading

BENDING SPOONS

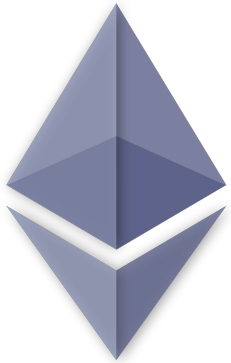
(2019 – current)

What I work on

Algorithmic trading systems

Focus: 👇
the cryptocurrency market(s)

What is Ethereum?



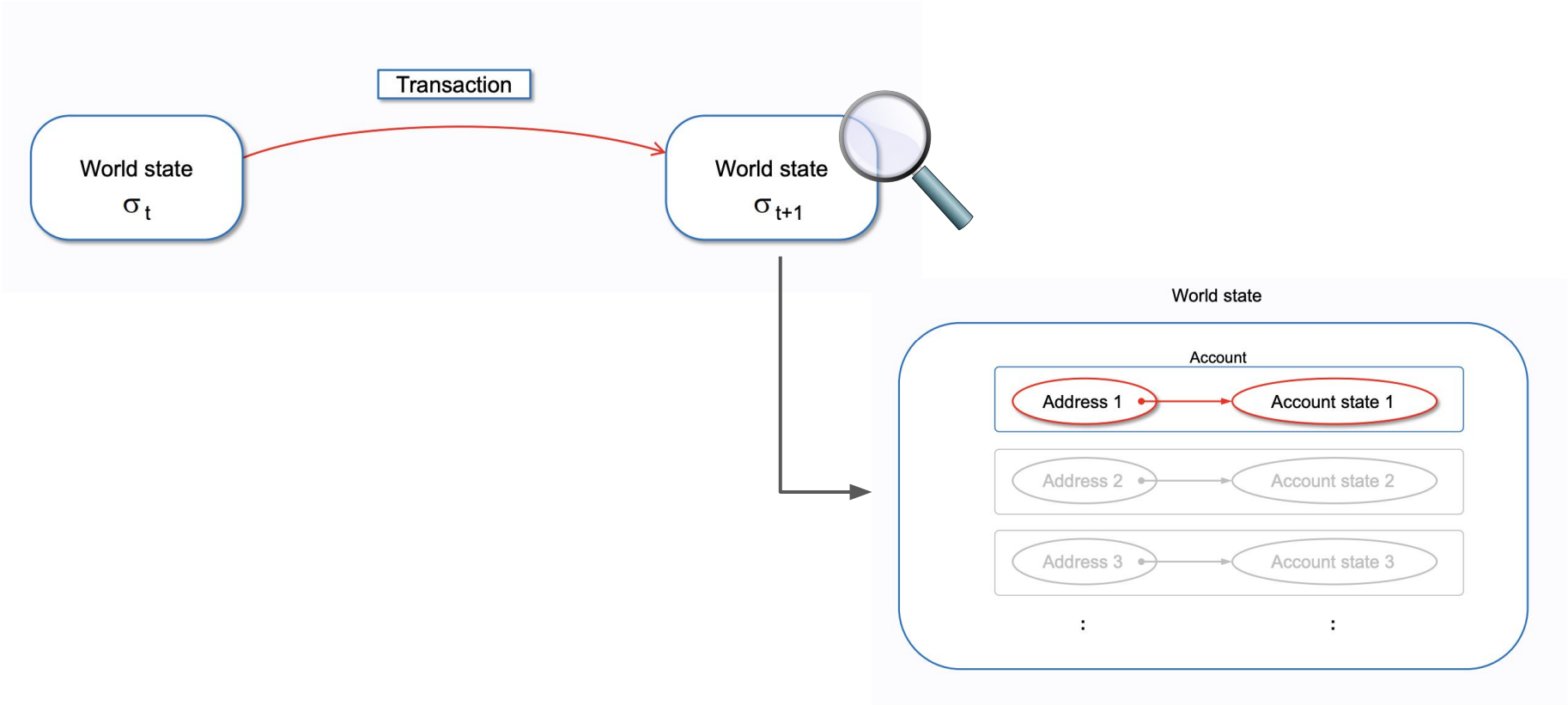
👉 **digital currency** (Ether / ETH)

- store and transfer value (like Bitcoin)
- (costly) payment method

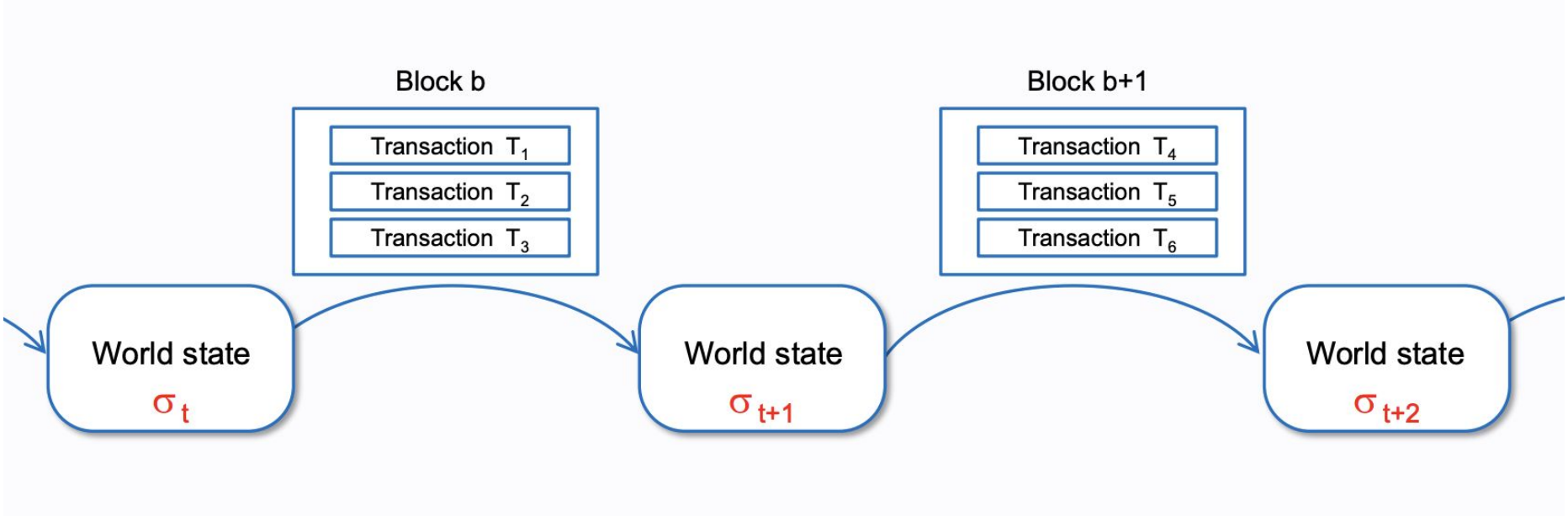
👉 deterministic, distributed **state machine**; “programmable money”

👉 **a platform of decentralized applications** (DApps)

The state machine

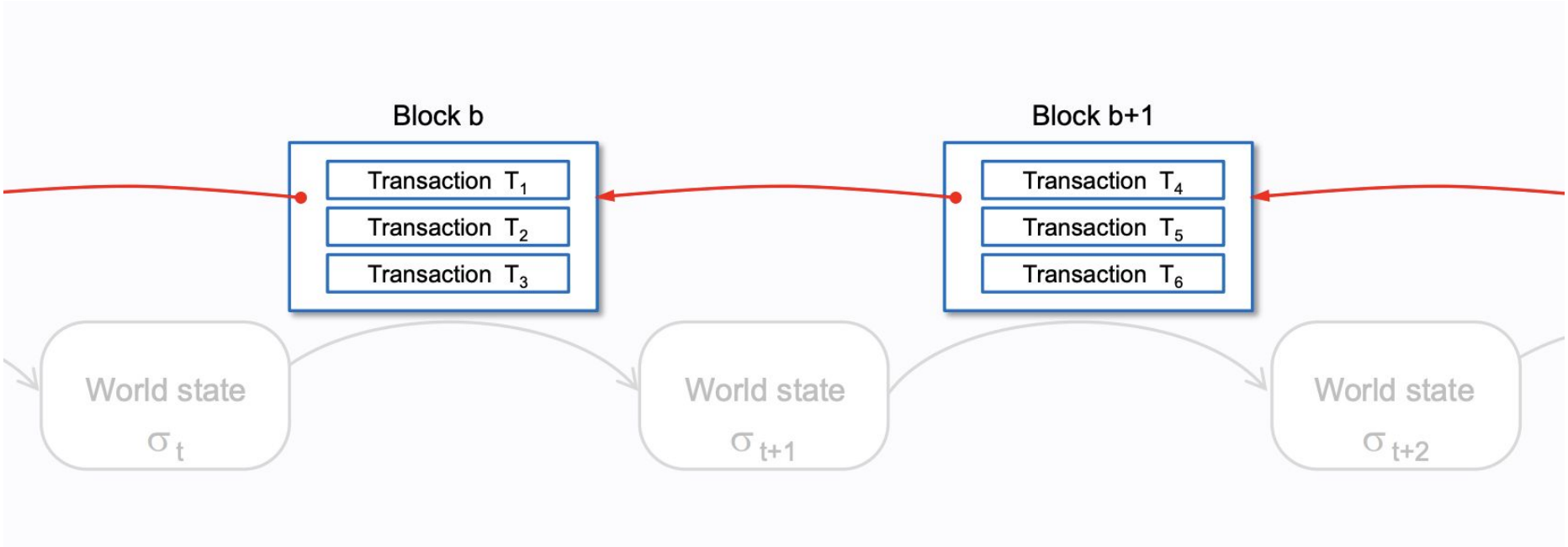


Transactions are arranged in “blocks”



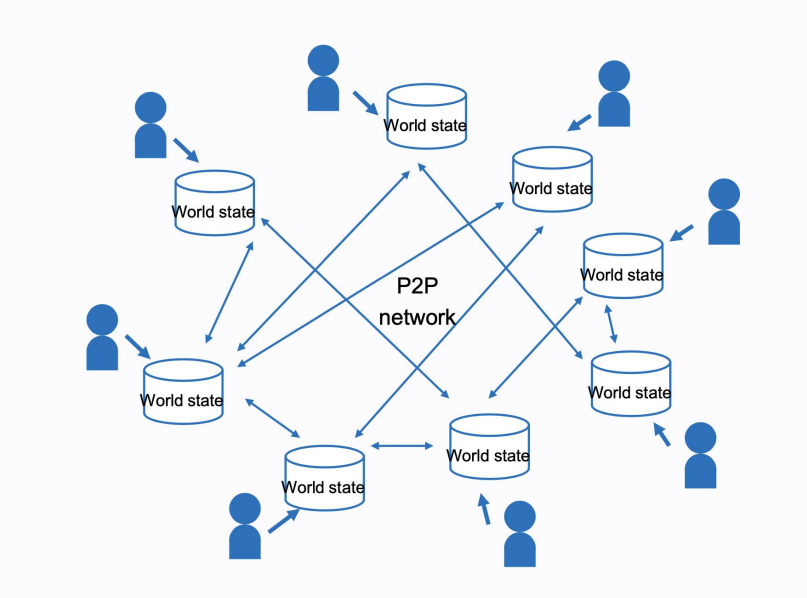
The **Ethereum Virtual Machine (EVM)** executes the transactions to compute the next logical state.

⛓ Chain of blocks = “blockchain”



👉 The state is distributed globally
👉 State changes are governed by the rule of consensus

🤝 Consensus on the Ethereum network



Ethereum is robust to:

► Partitioning
► Bad actors

👉🏼 Anyone can operate an Ethereum node.

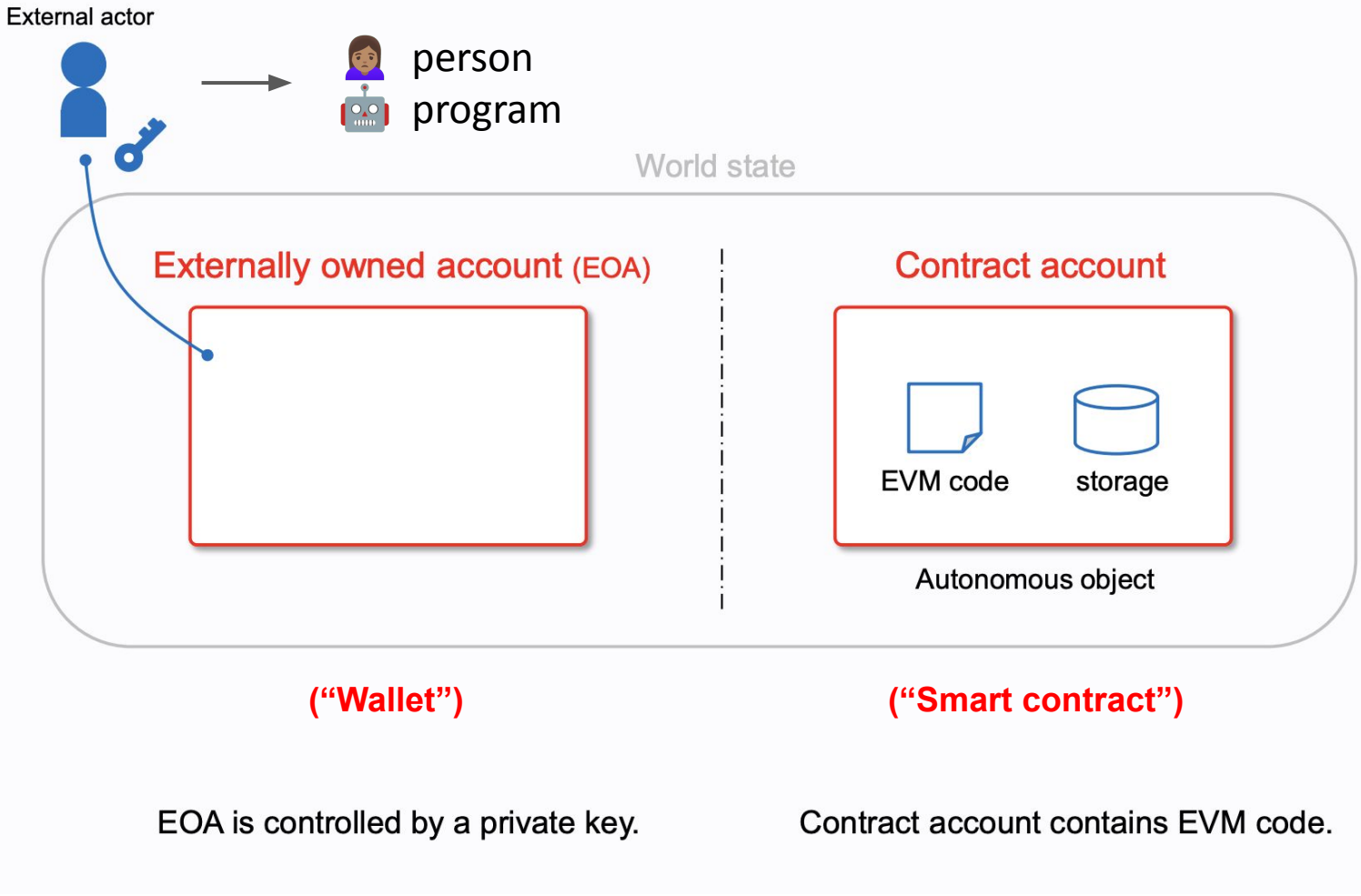
👉🏼 Each Ethereum node keeps track of the current world state (= confirmed blocks).

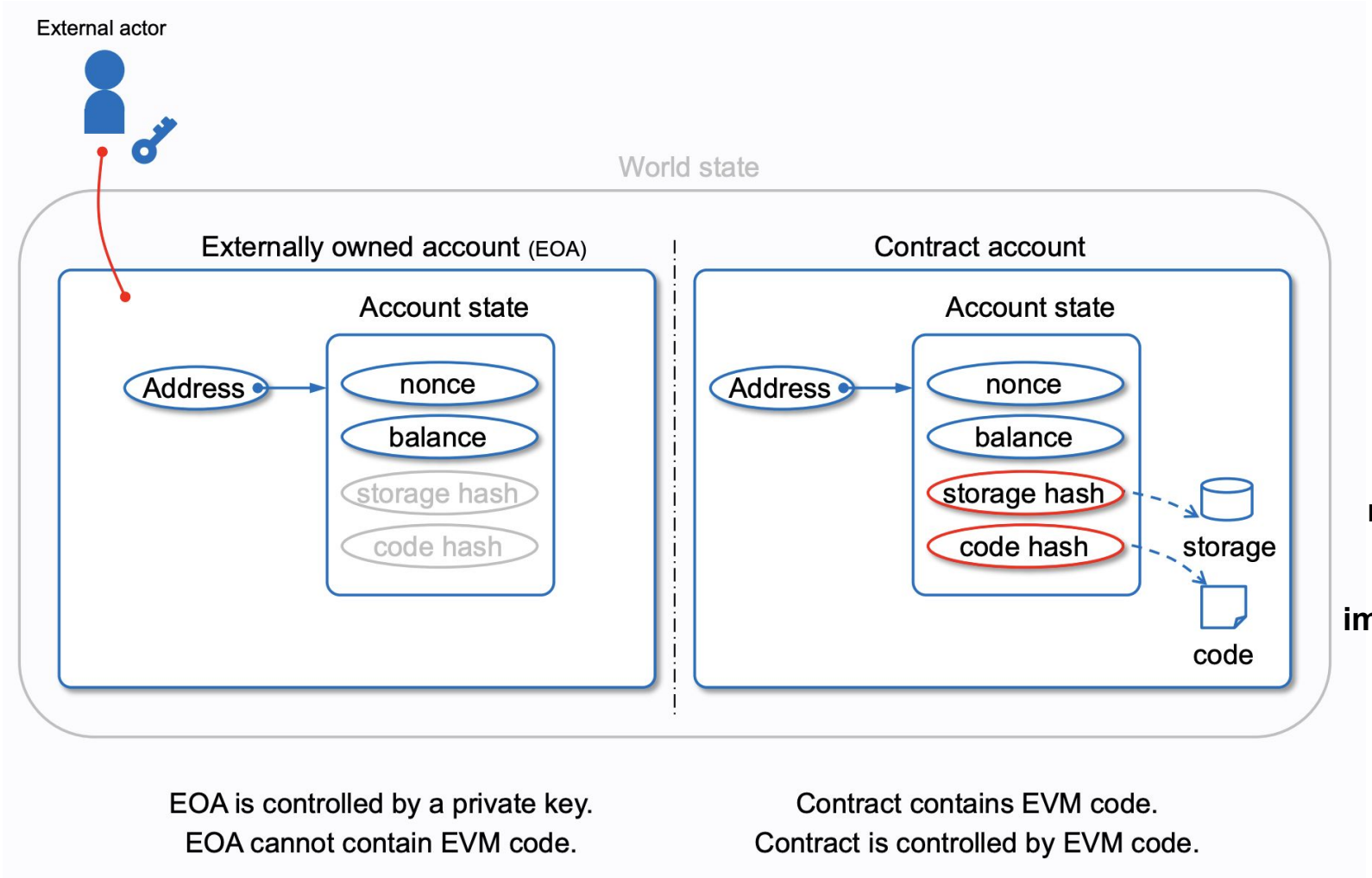
👉🏼 *Gossip protocol* to broadcast the unconfirmed transactions, to be included in a future block.

🥵 **Proof of work**: some Ethereum nodes (the “miners”) spend CPU time to “mine” the next block of transactions. Only one wins.

✅ All the other nodes can verify that the miner correctly signed a block.
Consensus is established:

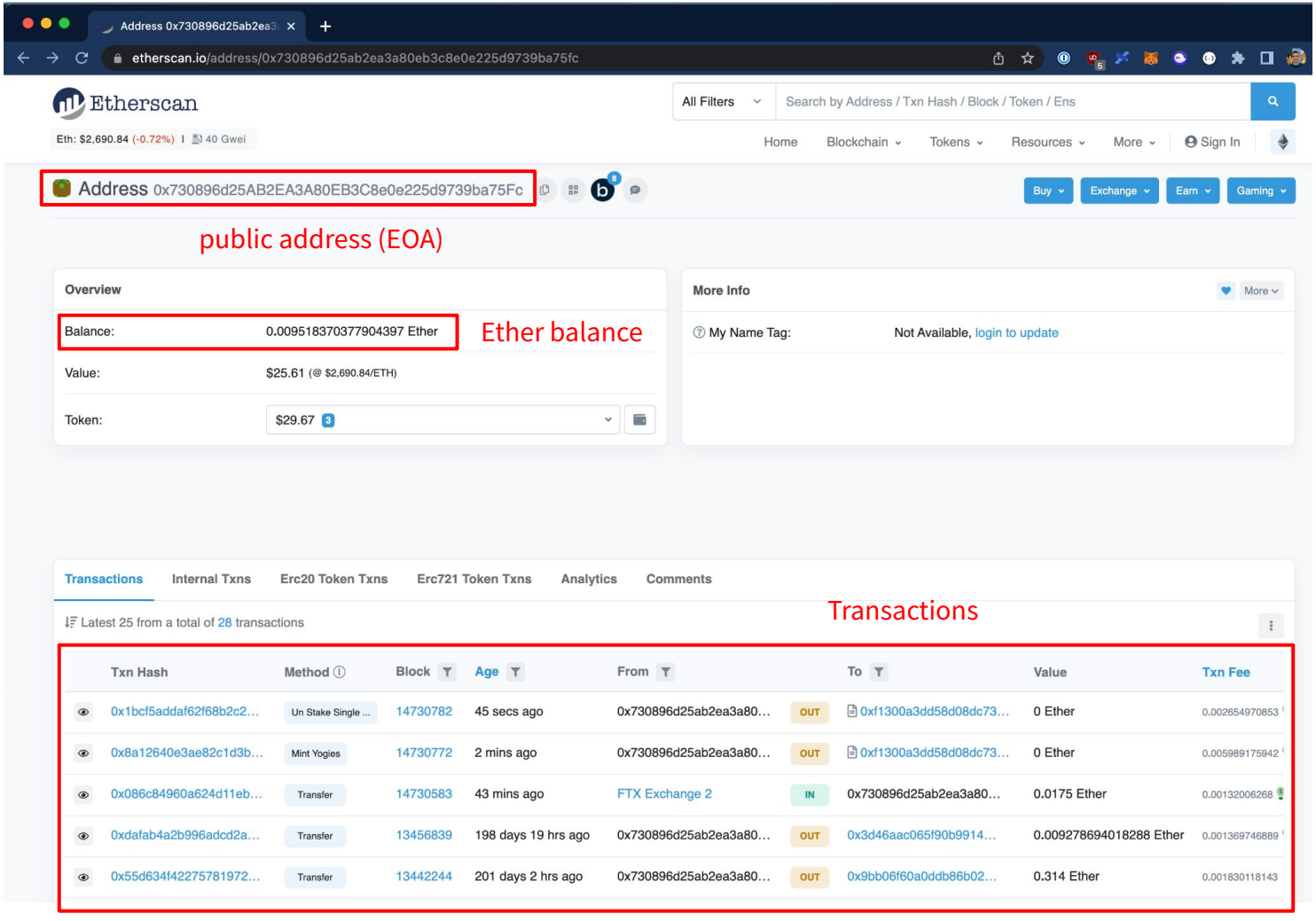
- current block’s transactions are added to world state
- miners start to work on the next block

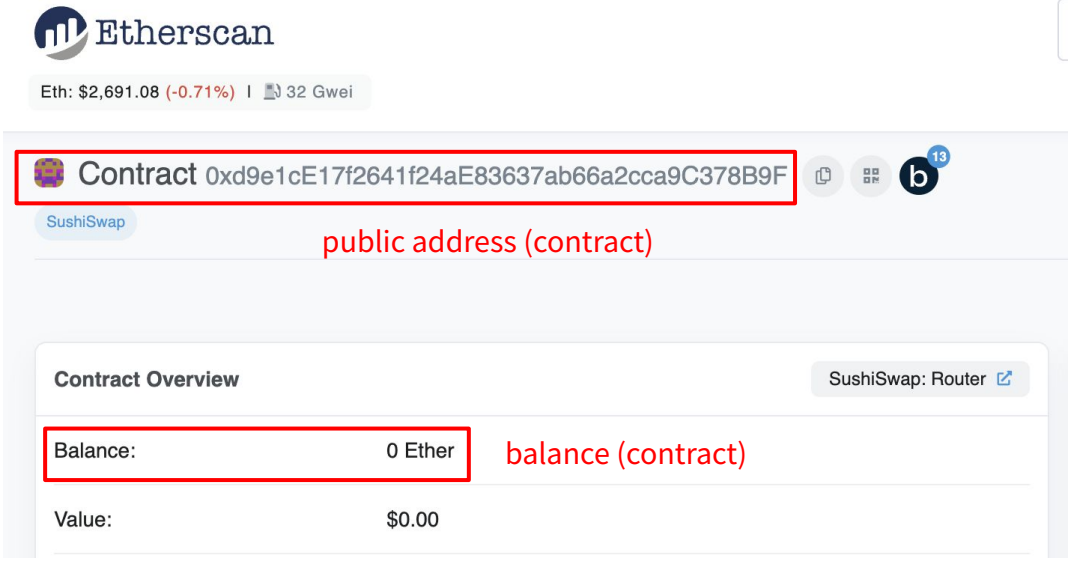

External actor
person
program
World state
Externally owned account (EOA)
Contract account
EVM code
storage
Autonomous object
("Wallet")
("Smart contract")
EOA is controlled by a private key.
Contract account contains EVM code.

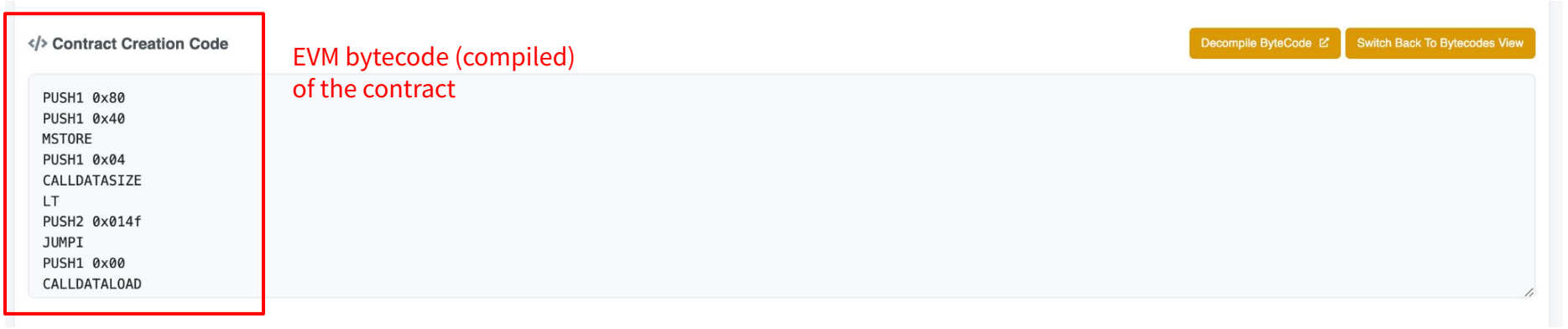

External actor
World state
Externally owned account (EOA)
Account state
Address
nonce
balance
storage hash
code hash
Contract account
Account state
Address
nonce
balance
storage hash
code hash
storage
code
mutable
immutable
EOA is controlled by a private key.
EOA cannot contain EVM code.
Contract contains EVM code.
Contract is controlled by EVM code.

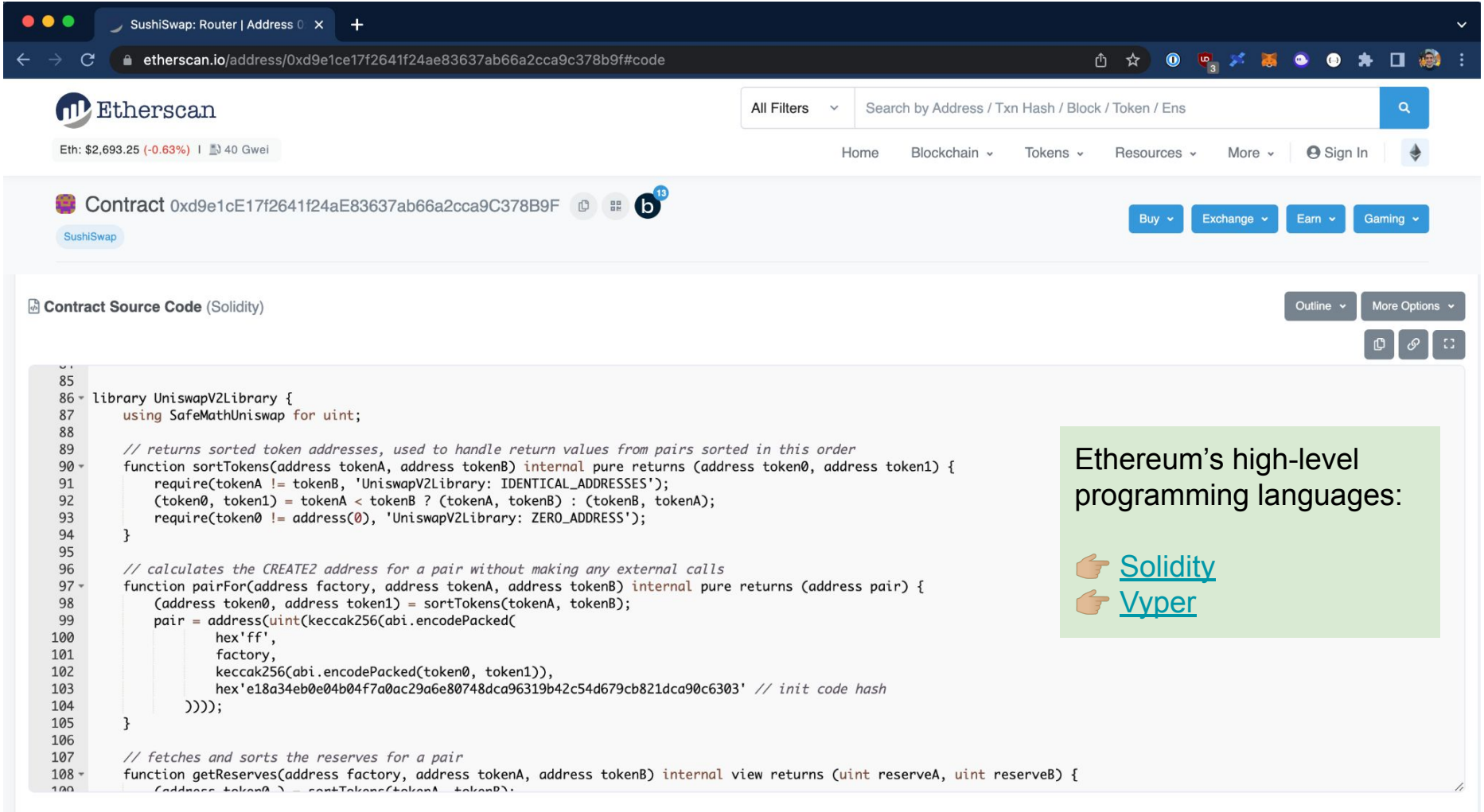
EOAs vs. contract account

	Externally Owned Account	Contract Account
Public address	✅	✅
Private key	✅	❌
Ether balance	✅	✅
Code (immutable)	❌	✅
Data storage (mutable)	❌	✅
Can initiate transactions	✅	❌


Address 0x730896d25ab2ea3
etherscan.io/address/0x730896d25ab2ea3a80eb3c8e0e225d9739ba75fc
Etherscan
All Filters
Search by Address / Txn Hash / Block / Token / Ens
Eth: $2,690.84 (-0.72%) | 40 Gwei
Home
Blockchain
Tokens
Resources
More
Sign In
Address 0x730896d25AB2EA3A80EB3C8e0e225d9739ba75Fc
Buy
Exchange
Earn
Gaming
public address (EOA)
Overview
Balance:
0.009518370377904397 Ether
Ether balance
Value:
$25.61 (@ $2,690.84/ETH)
Token:
$29.67
More Info
My Name Tag:
Not Available, login to update
Transactions
Internal Txns
Erc20 Token Txns
Erc721 Token Txns
Analytics
Comments
Latest 25 from a total of 28 transactions
Transactions
Txn Hash
Method
Block
Age
From
To
Value
Txn Fee
0x1bcf5addaf62f68b2c2...
Un Stake Single ...
14730782
45 secs ago
0x730896d25ab2ea3a80...
OUT
0xf1300a3dd58d08dc73...
0 Ether
0.002654970853
0x8a12640e3ae82c1d3b...
Mint Yogies
14730772
2 mins ago
0x730896d25ab2ea3a80...
OUT
0xf1300a3dd58d08dc73...
0 Ether
0.005989175942
0x086c84960a624d11eb...
Transfer
14730583
43 mins ago
FTX Exchange 2
IN
0x730896d25ab2ea3a80...
0.0175 Ether
0.00132006268
0xdafab4a2b996adcd2a...
Transfer
13456839
198 days 19 hrs ago
0x730896d25ab2ea3a80...
OUT
0x3d46aac065f90b9914...
0.009278694018288 Ether
0.001369746889
0x55d634f42275781972...
Transfer
13442244
201 days 2 hrs ago
0x730896d25ab2ea3a80...
OUT
0x9bb06f60a0ddb86b02...
0.314 Ether
0.001830118143


Etherscan
Eth: $2,691.08 (-0.71%) | 32 Gwei
Contract 0xd9e1cE17f2641f24aE83637ab66a2cca9C378B9F
SushiSwap
public address (contract)
Contract Overview
SushiSwap: Router
Balance: 0 Ether
balance (contract)
Value: $0.00


Contract Creation Code
EVM bytecode (compiled)
of the contract
Decompile ByteCode
Switch Back To Bytecodes View
PUSH1 0x80
PUSH1 0x40
MSTORE
PUSH1 0x04
CALLDATASIZE
LT
PUSH2 0x014f
JUMPI
PUSH1 0x00
CALLDATALOAD


SushiSwap: Router | Address 0
etherscan.io/address/0xd9e1ce17f2641f24ae83637ab66a2cca9c378b9f#code
Etherscan
All Filters
Search by Address / Txn Hash / Block / Token / Ens
Eth: $2,693.25 (-0.63%) | 40 Gwei
Home
Blockchain
Tokens
Resources
More
Sign In
Contract 0xd9e1cE17f2641f24aE83637ab66a2cca9C378B9F
SushiSwap
Buy
Exchange
Earn
Gaming
Contract Source Code (Solidity)
Outline
More Options
85
86 library UniswapV2Library {
87 using SafeMathUniswap for uint;
88
89 // returns sorted token addresses, used to handle return values from pairs sorted in this order
90 function sortTokens(address tokenA, address tokenB) internal pure returns (address token0, address token1) {
91 require(tokenA != tokenB, 'UniswapV2Library: IDENTICAL_ADDRESSES');
92 (token0, token1) = tokenA < tokenB ? (tokenA, tokenB) : (tokenB, tokenA);
93 require(token0 != address(0), 'UniswapV2Library: ZERO_ADDRESS');
94 }
95
96 // calculates the CREATE2 address for a pair without making any external calls
97 function pairFor(address factory, address tokenA, address tokenB) internal pure returns (address pair) {
98 (address token0, address token1) = sortTokens(tokenA, tokenB);
99 pair = address(uint(keccak256(abi.encodePacked(
100 hex'ff',
101 factory,
102 keccak256(abi.encodePacked(token0, token1)),
103 hex'e18a34eb0e04b04f7a0ac29a6e80748dca96319b42c54d679cb821dca90c6303' // init code hash
104))));
105 }
106
107 // fetches and sorts the reserves for a pair
108 function getReserves(address factory, address tokenA, address tokenB) internal view returns (uint reserveA, uint reserveB) {
Ethereum's high-level programming languages:
Solidity
Vyper

Transactions

For 0xd9e1ce17f2641f24ae83637ab66a2cca9c378b9f SushiSwap: Router

A total of 3,710,354 transactions found

(Showing the last 100k records)

First | Page 1 of 2000 | Last

Txn Hash	Method	Block	Age	From		To	Value	Txn Fee
0x3e7e01f9c3b3f04f8c9…	Swap Exact Token...	14730841	32 secs ago	0xe70758e913bf06e8a8…	IN	SushiSwap: Router	0 Ether	0.00525055
0x64f6dbf8f367b3c2d50…	Swap Exact ETH F...	14730835	2 mins ago	0xef956d6c8f457aa7be2…	IN	SushiSwap: Router	0.00417 Ether	0.00560116
0x86345fb4583442c7d7…	Swap Exact ETH F...	14730830	3 mins ago	dolomode.eth	IN	SushiSwap: Router	0.36 Ether	0.00461736
0x962c73ec1bf1569729…	Swap Exact ETH F...	14730830	3 mins ago	rickeynft.eth	IN	SushiSwap: Router	1.8 Ether	0.00461788
0x6f1aaf08d96a0f24ac5…	Swap Exact Token...	14730830	3 mins ago	0xdfa421c338345c9248…	IN	SushiSwap: Router	0 Ether	0.00821112
0x0fe05877fb595dc1acd…	Swap Exact Token...	14730822	4 mins ago	0x3cbb2a1eff7860d0aeb…	IN	SushiSwap: Router	0 Ether	0.00922256
0x17d5f2aea198fcfa4b2…	Swap Tokens For ...	14730814	7 mins ago	0xf07704777d6bc182bf2…	IN	SushiSwap: Router	0 Ether	0.00529443
0x9a7f695996a9f4d639d…	Swap ETH For Exa...	14730812	7 mins ago	0xc18322da31df55cf9a8…	IN	SushiSwap: Router	0.025273482285692 Ether	0.00604031

API of a smart contract: **functions** callable from EOAs or other contracts.

Function types

Pure: does not read nor write the state

View: does not write the state

Public: can be called by transactions

- 👉 from other contracts
- 👉 from EOAs directly

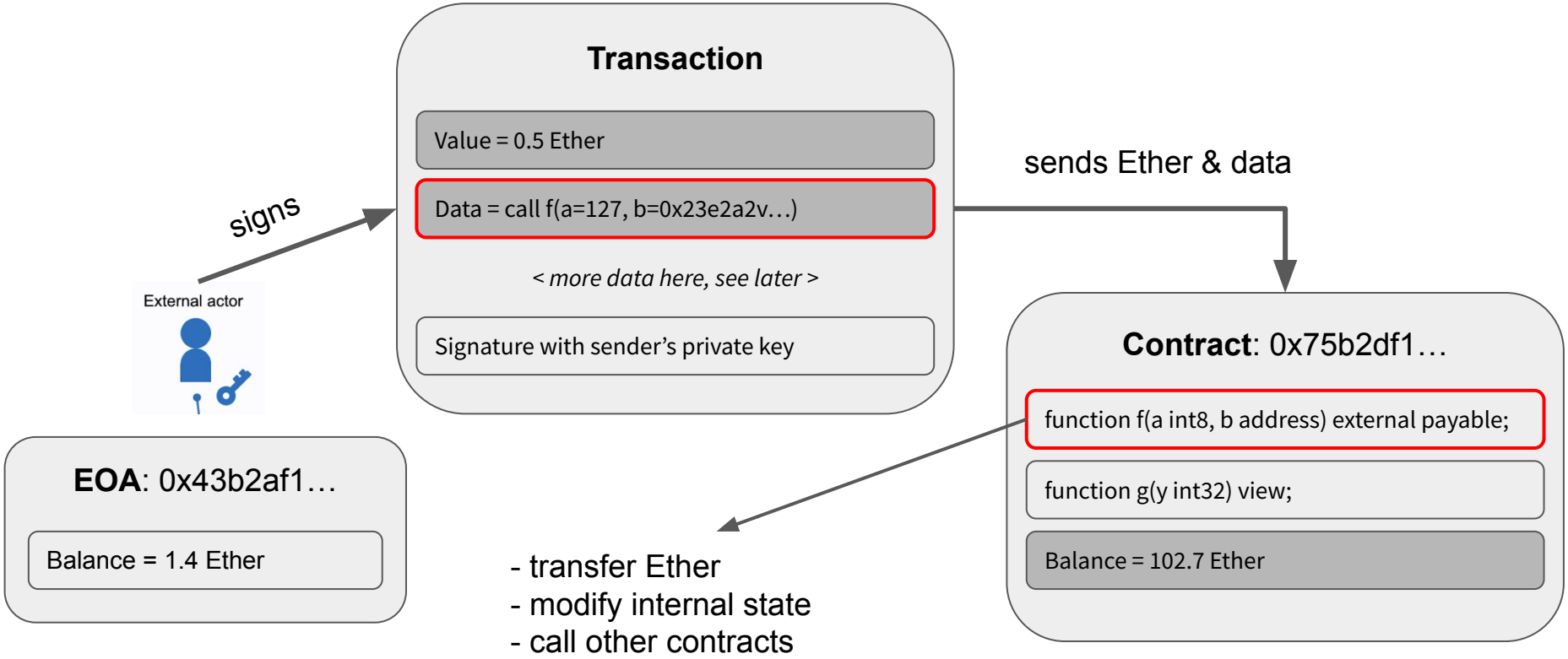
Private: can be called only by the contract itself

Payable: the function can accept Ether

Contract Source Code (Solidity)

```
8
9  interface IUniswapV2Pair {
10     event Approval(address indexed owner, address indexed spender, uint value);
11     event Transfer(address indexed from, address indexed to, uint value);
12
13     function name() external pure returns (string memory);
14     function symbol() external pure returns (string memory);
15     function decimals() external pure returns (uint8);
16     function totalSupply() external view returns (uint);
17     function balanceOf(address owner) external view returns (uint);
18     function allowance(address owner, address spender) external view returns (uint);
19
20     function approve(address spender, uint value) external returns (bool);
21     function transfer(address to, uint value) external returns (bool);
22     function transferFrom(address from, address to, uint value) external returns (bool);
23
24     function DOMAIN_SEPARATOR() external view returns (bytes32);
25     function PERMIT_TYPEHASH() external pure returns (bytes32);
26     function nonces(address owner) external view returns (uint);
27
28     function permit(address owner, address spender, uint value, uint deadline, uint8 v, byt
29
30     event Mint(address indexed sender, uint amount0, uint amount1);
31     event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);
32     event Swap(
```

Smart contract transaction: a simplified example



The EVM as a Turing machine

👉🏽 Ethereum contract code is **Turing-complete**: it can implement any computable function.

⚠️ This makes Ethereum a *general-purpose* global distributed computer.

👉🏽 Therefore, the **E**thereum **V**irtual **M**achine is equivalent to a **Universal Turing Machine**:

Input tape: transactions, containing data + Ether

Code: the smart contracts' function(s) to be executed

State: the set of all smart contracts' states and account balances

Output tape ("side effect"): transfer of Ether across accounts

All good, but…

WILL IT HALT?

Problem #1: smart contract termination ⚡

In computability theory, the **halting problem** is the problem of **determining, from a description of an arbitrary computer program and an input, whether the program will finish running, or continue to run forever**.

Alan Turing proved in 1936 that **a general algorithm** to solve the halting problem for all possible program-input pairs **cannot exist**.

👉 The EVM is Turing-complete

⚠️ If a smart contract runs forever, **the EVM gets stuck** (= unusable!)

❌ No way to detect and reject not-halting (or expensive) smart contract calls!

Gas ⛽

Ethereum's solution: an **economic disincentive**

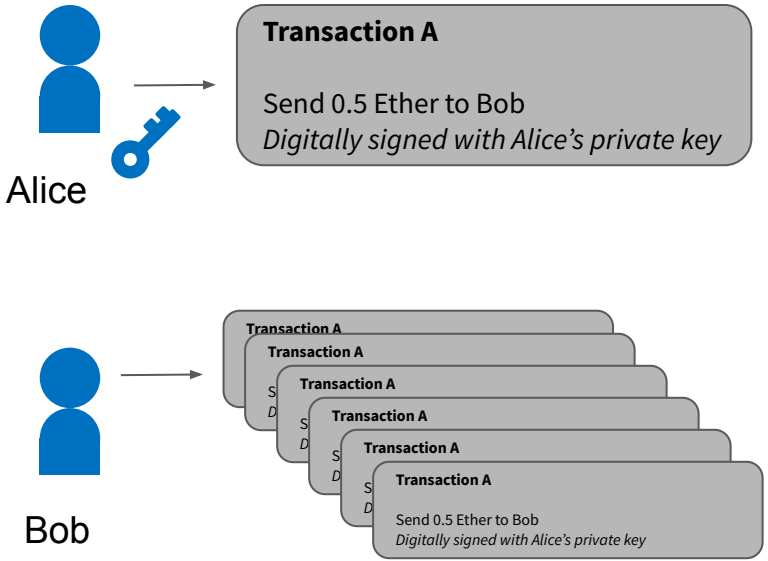
How to avoid that users abuse the EVM with long-running programs?
How does Ethereum guarantee termination?

- 👉 Each EVM opcode costs ***gas*** to execute
- ✋🏾 Users set a *gas limit* in each transaction
- 💸 The more code you run, the more you pay (up to the gas limit!)
- ⛔ Gas limit is reached → transaction is terminated and reverted
  - 💸 Only effect: the user pays the *gas limit* in full.

Problem #2: replay attacks 👯

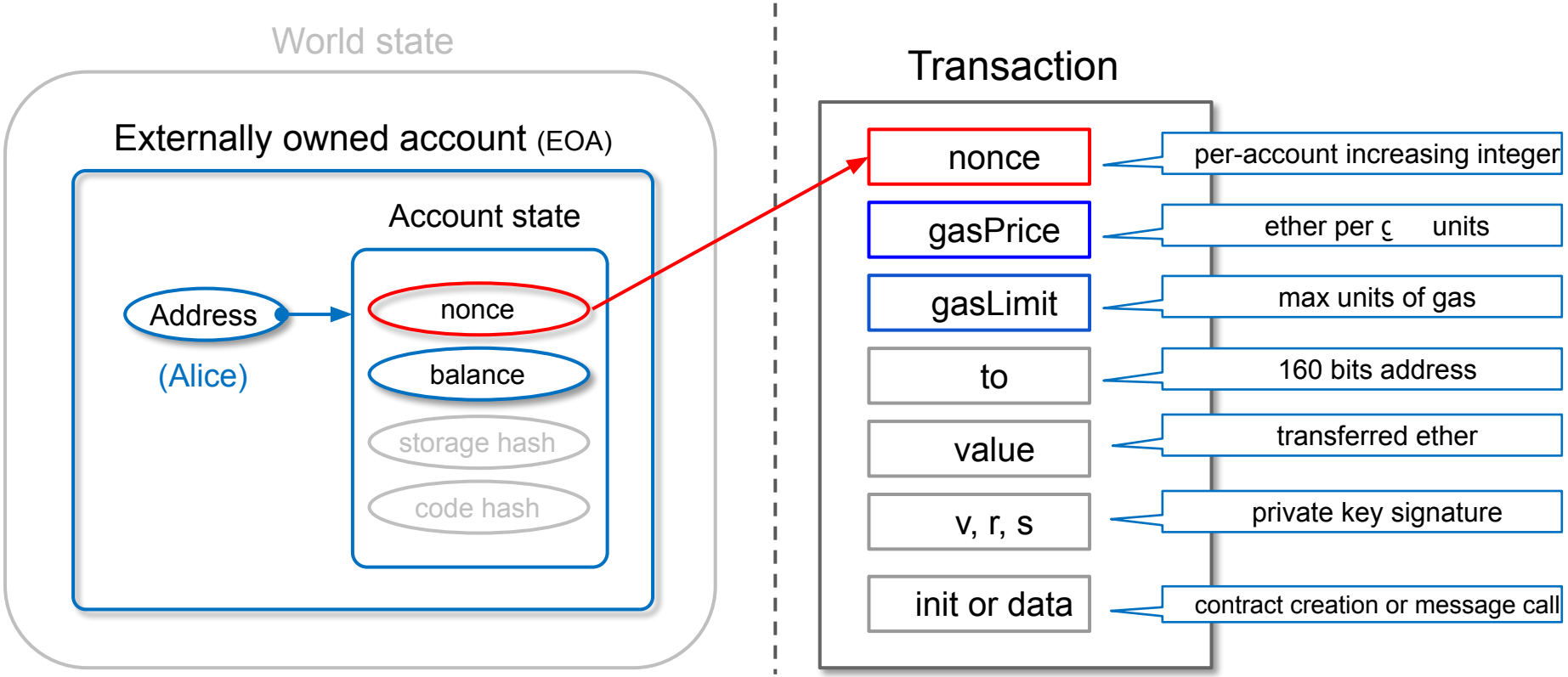
What is stopping people from *replaying* a transaction?

1. Alice signs a valid transaction:
   “Send 0.5 Ether to Bob”
2. The transaction is executed.
3. Bob reads the transaction from the public blockchain and **resends it to the network**.
4. *Profit?*



Nonce: A scalar value equal to the number of transactions sent from this address

Nonce: A scalar value equal to the number of transactions sent from this address



Smart contract example: ERC20 Tokens

USDC, USDT, SHIB, DAI, … how to create a new **token** (“coin”) on Ethereum?
You must implement the ERC20 *interface.*

Note: Ether (the native currency) is not an ERC20 token! (Wrapped Ether (WETH) is.)

```
interface IERC20 {
   /* Returns the amount of tokens in existence. */
   function totalSupply() external view returns (uint256);

   /* Returns the amount of tokens owned by `account`. */
   function balanceOf(address account) external view returns (uint256);

   /* Moves `amount` tokens from the caller's account to `to`.
    * Returns a boolean value indicating whether the operation succeeded. */
   function transfer(address to, uint256 amount) external returns (bool);

   /* … more functions not shown here for brevity */
}
```

⚙️ ERC20 reference implementation (link)

```
contract ERC20 is IERC20 {
   mapping(address => uint256) private _balances;

   uint256 private _totalSupply;
   string private _name;
   string private _symbol;

   function balanceOf(address account) public view virtual override returns (uint256) {
       return _balances[account];
   }

   function transfer(address to, uint256 amount) public virtual override returns (bool) {
       address owner = msg.sender;
       uint256 fromBalance = _balances[owner];
       require(fromBalance >= amount, "ERC20: transfer amount exceeds balance");

       _balances[from] = fromBalance - amount;
       _balances[to] += amount;

       return true;
   }
   /* ... more implementation ...*
}
```

🚨 ERC20 issues

For the curious:
[USDT (Tether) contract code]

Ether is needed	To transfer ERC20 tokens, you need to pay transaction fees—using Ether.
Multiple implementations	Feature *and* bug. ERC20 is a mere interface: tokens can extend it.
Vulnerable implementations	Custom implementation may have exploitable bugs → assets at risk!
Malicious implementations	👉 Increase `totalSupply()` (inflationary token) 👉 `transfer()` to "wrong" destination 👉 "Blacklisted", frozen addresses 👉 Backdoors, rug pulls, "owner" accounts…

ERC20 tokens are *just code*.
⚠️ Careful when interacting with unknown / untrusted ERC20 tokens on-chain!

DApp example: Uniswap

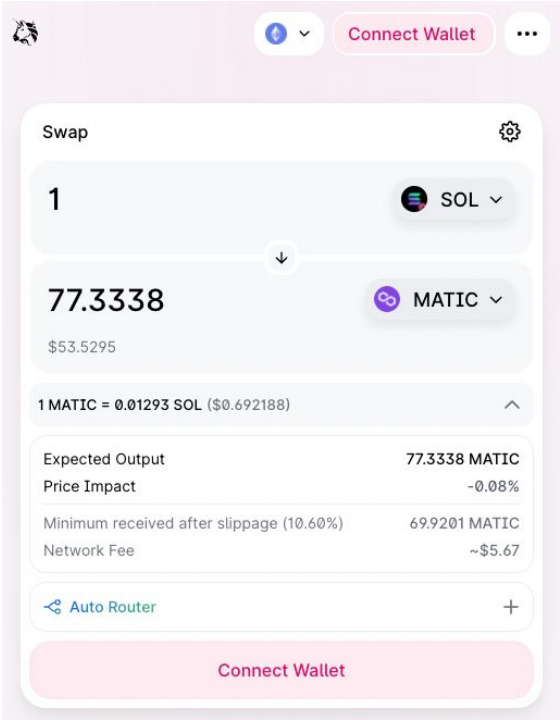
DApp = Decentralized App

Wallet + web frontend + smart contract(s)

Uniswap: a decentralized exchange (DEX).

Swap ERC20 tokens without a central authority. Smart contracts execute the swaps directly on the users' wallets.

https://app.uniswap.org/



🎮 DApp example: play-to-earn in the metaverse

Metaverse: a universal, immersive **virtual world**, facilitated by the use of AR and VR headsets.

Non-Fungible Token (ERC 721): a smart contract implementing a “unique”, collectible, transferable item.

Examples: lottery tickets, art, memes, event passes

Play-to-earn game: a MMORPG where users exchange economic value through the blockchain.

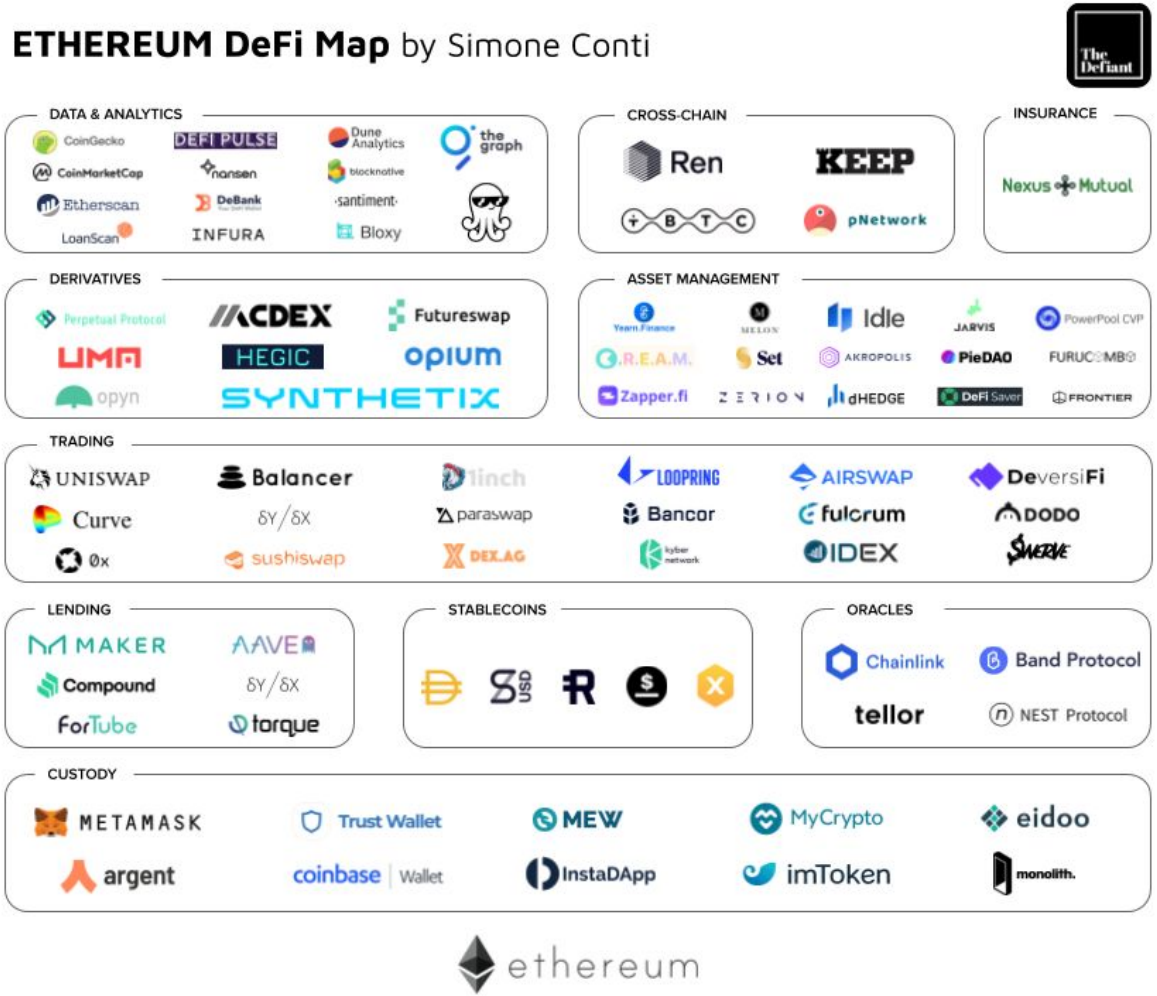
⚠️ Faster, cheaper blockchains are typically used in place of Ethereum.

Game “currency” (coins, resources)	ERC-20 token (fungible)
Collectibles (equipment, armor, cards, medals…)	ERC-721 token (non-fungible)

DeFi

=

Decentralized Finance



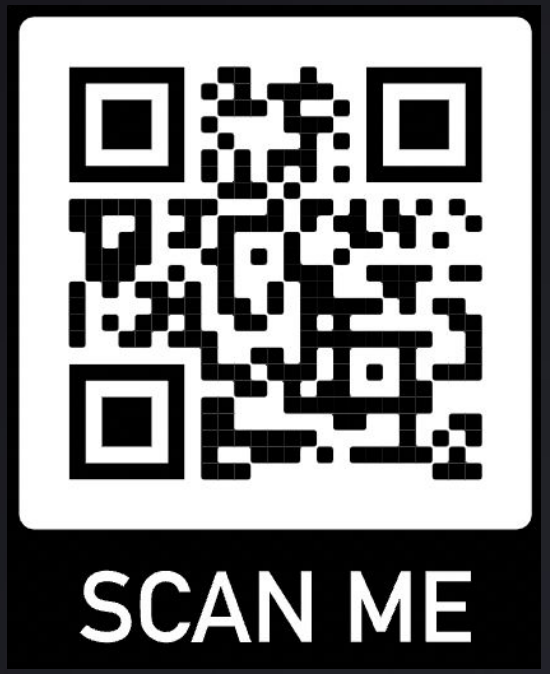
Source

Conclusion

- Ethereum: a distributed FSM
- Consensus mechanism
- Accounts and contracts
- Transactions
- Termination and gas fees
- Replay attacks and nonces
- ERC20 tokens
- Applications

References

- Antonopoulos, Wood (2018) *Mastering Ethereum*
- Takenobu Tani (2018) *Ethereum EVM Illustrated*
- Luca Boiardi (2022) *Ethereum, cos'è e come funziona*
- Vitalik Buterin (2014) *Ethereum Whitepaper*



We’re hiring!

https://bndspn.com/Uni-careers

Contact me: pd@bendingspoons.com