Mr A.Dekhinet
University of BATNA 2 / Computer Science Department
a.dekhinet@univ-batna2.dz
http://staff.univ-batna2.dz/dekhinet-abdelhamid

Chapter 4 :
dApp : Decentralized Application
Ethereum & Smart Contract

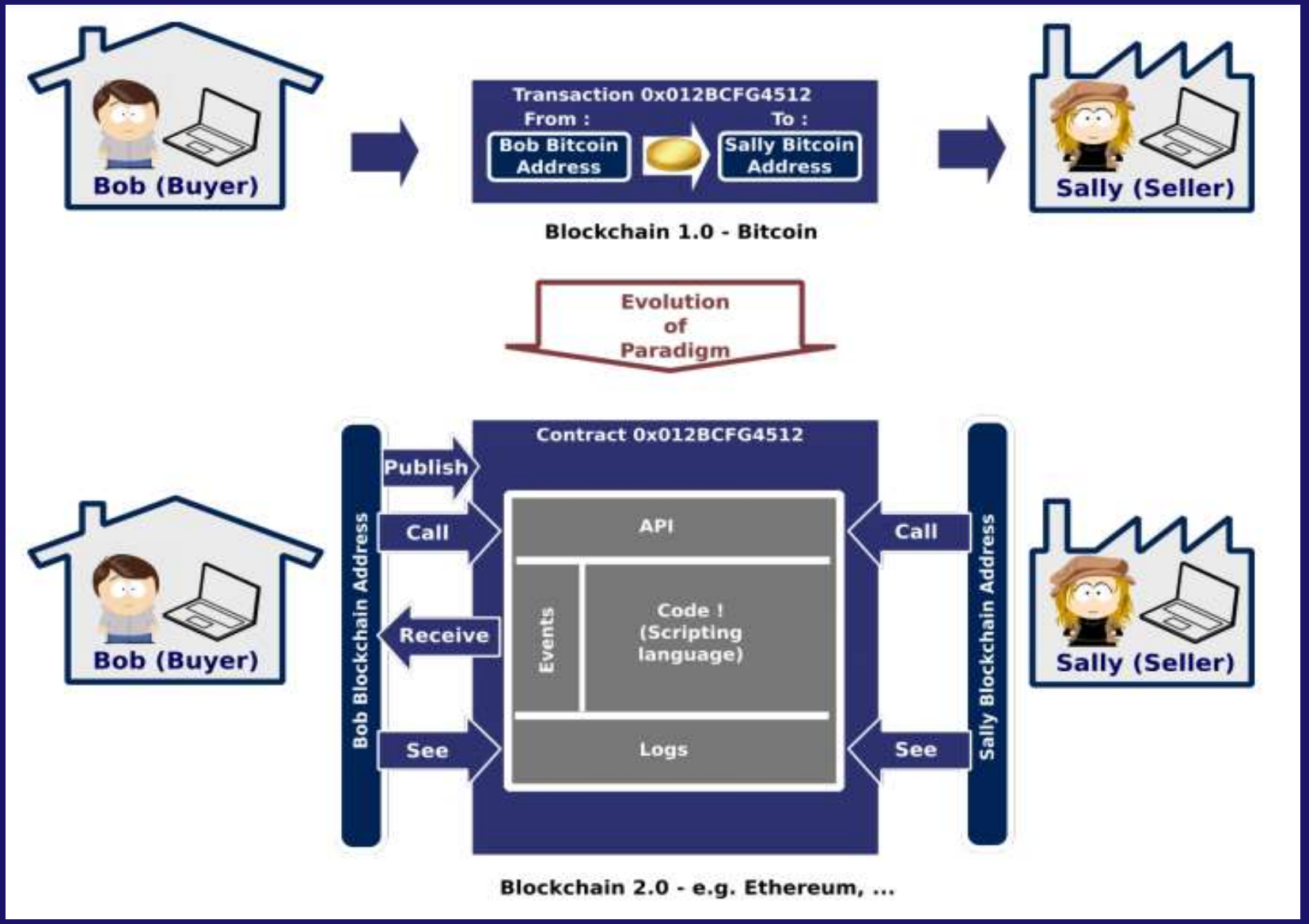
Ethereum

- Ethereum is a public blockchain based decentralized and distributed computing platform
- Proposed in late 2013 by Vitalik Buterin (cryptocurrency researcher and programmer)
- Not just money : Asset issuance, gambling, IoT, voting, supply chain, document verification, hundreds of applications
- Cryptocurrency : Ether (BTC fo BC)
- Ethereum Consensus : PoW (idem for BC)
- Hash function : Ethash (SHA 256 for BC)
- Blocktime : 14 seconds (10 minutes for BC)

Ethereum

- Ethereum platform is a general purpose Blockchain or programmable blockchain : Generalised Technology, in opposite to Bitcoin which is dedicated to manage cryptocurrency
- Ethereum is often described and interpreted as a distributed global computer "The world computer"
- Programmable blockchain
  - ↳ Allows users to develop their own decentralized applications (dApps)
- Ethereum provides support for Turing-complete language for application development
- Transactions record code, variables, the results of function calls, …
- Transactions are generated by a computer program : Smart Contract
- Application can be written in several languages, ranging from general-purpose languages like Java, C++, Python, JavaScript, Golang to platform-specific languages like Solidity for Ethereum or Bitcoin Script for Bitcoin.

Ethereum


Bob (Buyer)
Transaction 0x012BCFG4512
From :
To :
Bob Bitcoin Address
Sally Bitcoin Address
Sally (Seller)
Blockchain 1.0 - Bitcoin
Evolution of Paradigm
Contract 0x012BCFG4512
Publish
Call
Receive
See
API
Events
Code ! (Scripting language)
Logs
Bob Blockchain Address
Call
See
Sally Blockchain Address
Bob (Buyer)
Sally (Seller)
Blockchain 2.0 - e.g. Ethereum, ...

Ethereum

- For execution, Ethereum provides a trusted and decentralised virtual machine as runtime environment : Ethereum Virtual Machine (EVM).
- EVM is a stack based processor.

Let's summarize

Ethereum : Programmable blockchain platform (Open Source)

Solidity : Programming langage

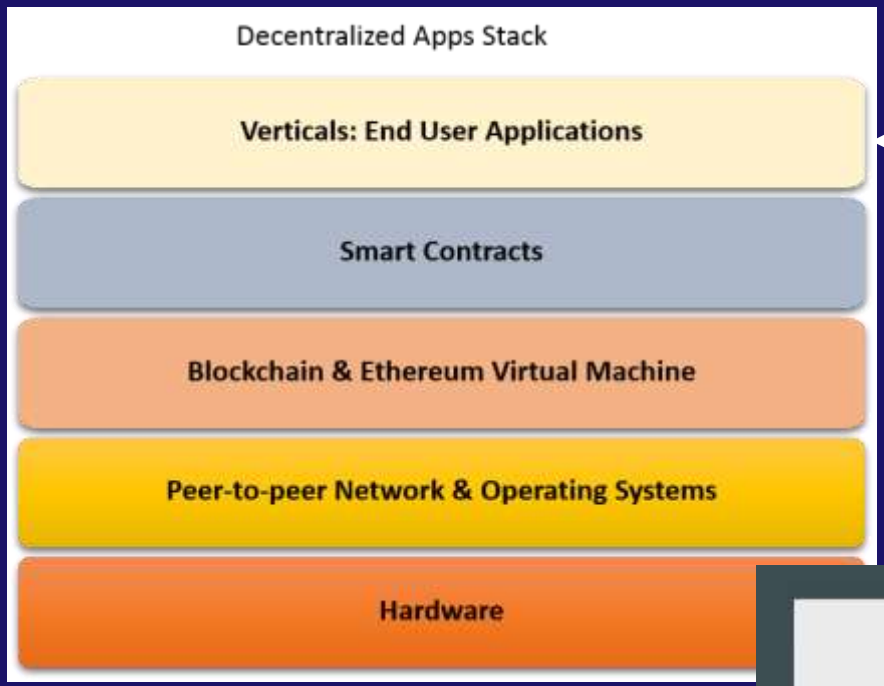
Smart Contract : Program

EVM : Execution runtime for Smart Contract writen in Solidity

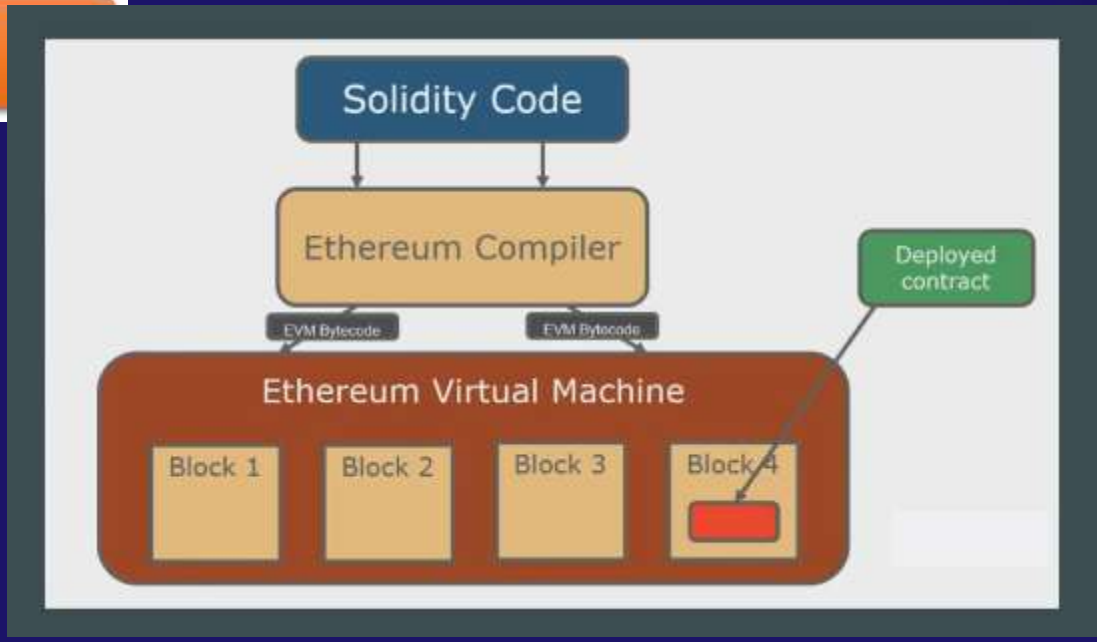
Compilation : Smart Contract is comipled into EVM assembly Byte Code

- To learn solidity go to **https://remix.ethereum.org** and you can start programming smart contracts without having to create your own Ethereum network platform.

Ethereum : Single Node

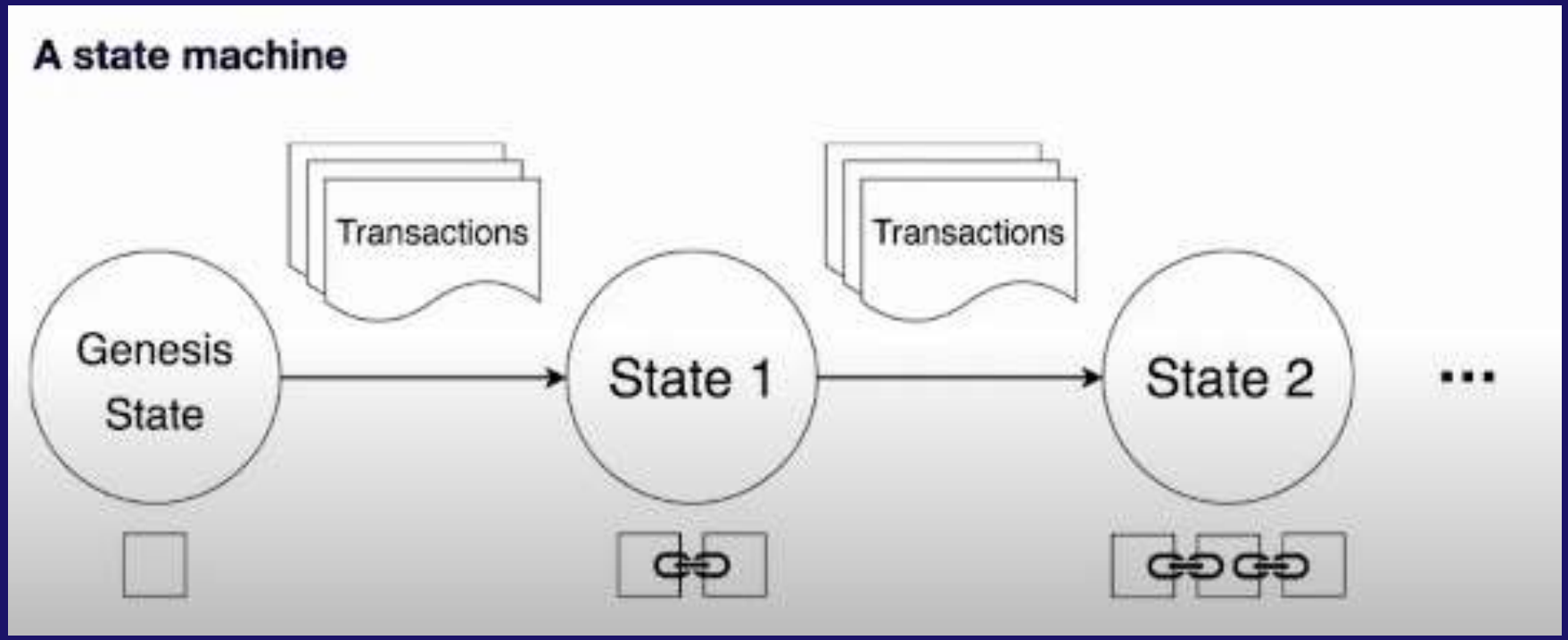


Web development : JS, HTML, …



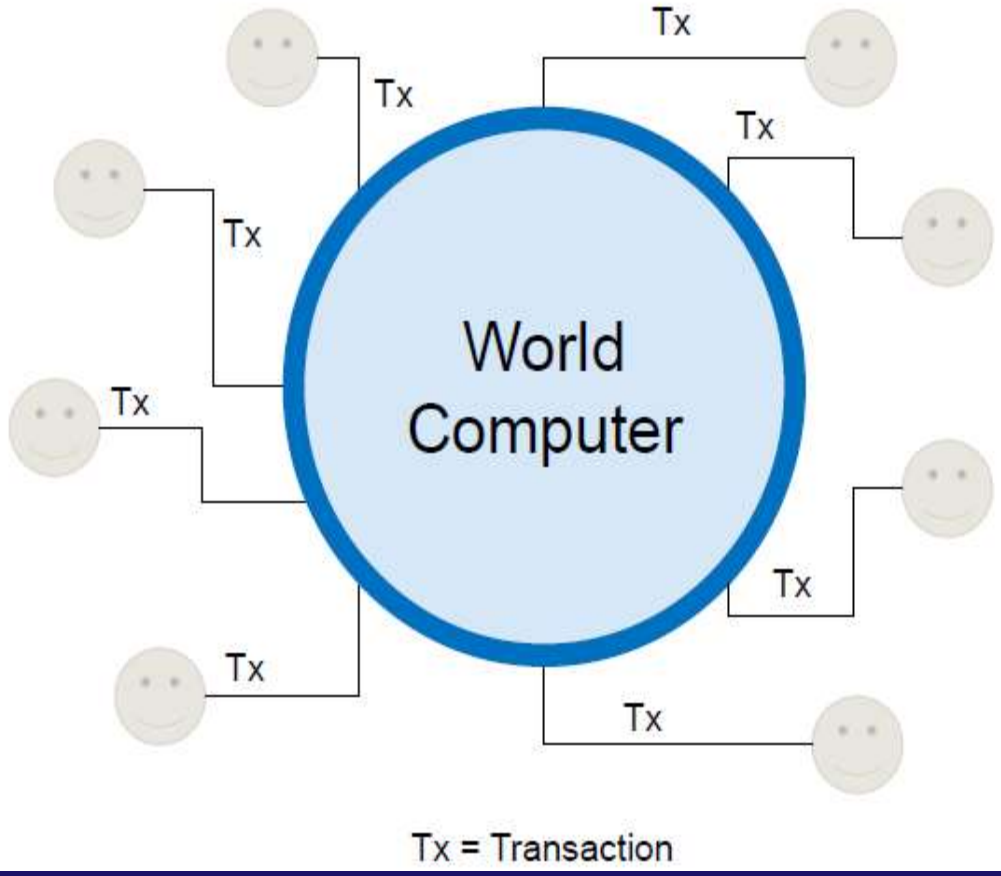
Ethereum

- From a computer science perspective, Ethereum is a deterministic but practically unbounded state machine, consisting of a globally accessible singleton state and a virtual machine that applies changes to that state
- Ethereum can be perceived as a distributed transaction-based state machine



The concept of a world computer

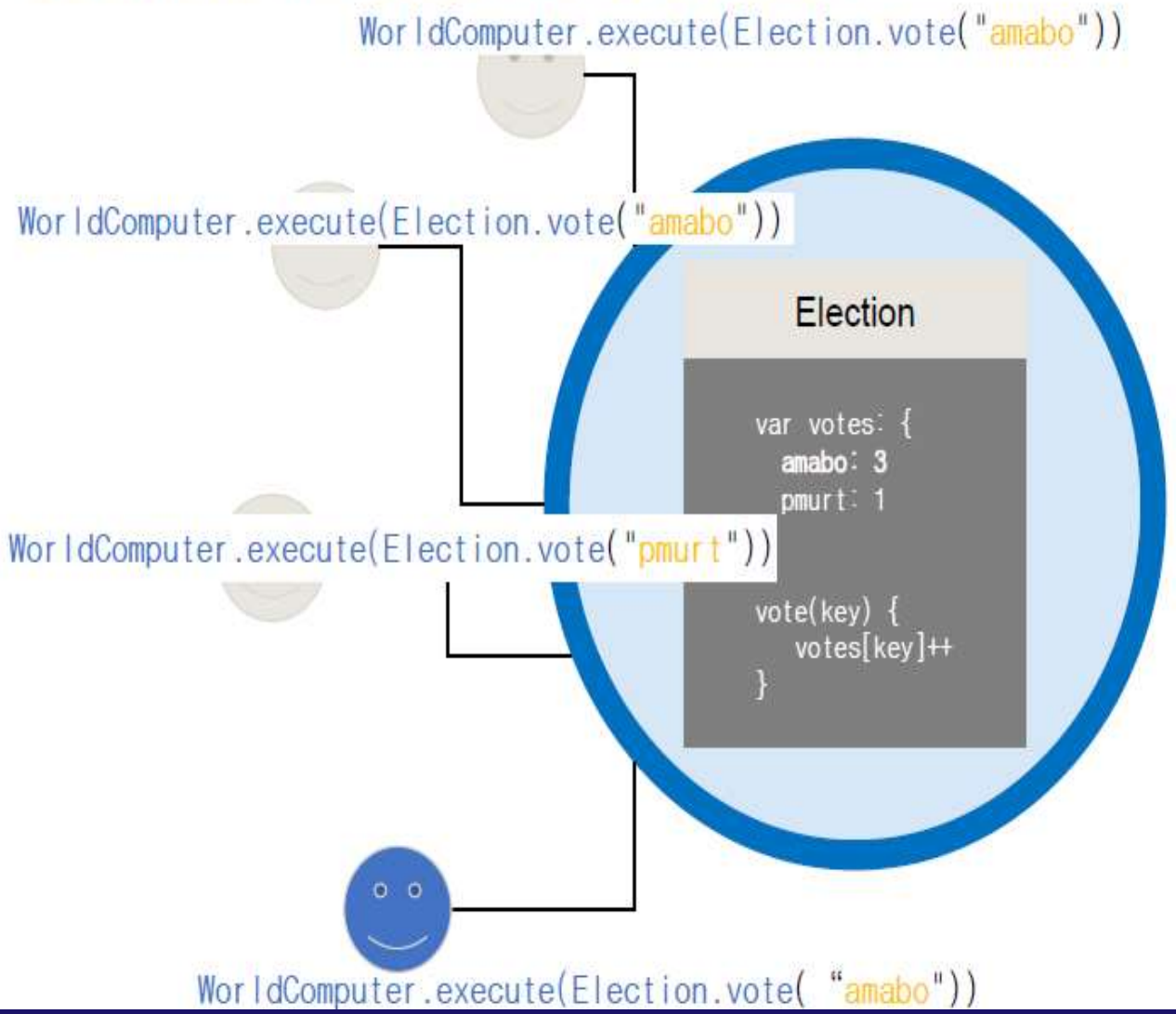
State Machine



Properties

- **All participants** are **using** the **same computer**
- Users issue **transactions to call programs** on the computer
- **Everyone shares** the same **resources** and **storage**
- The **computer has no explicit**, single **owner**
- **Using** the computer's **resources costs money**

Election example using a world computer



State of the world

State 0 (initial)
(nothing happened yet)

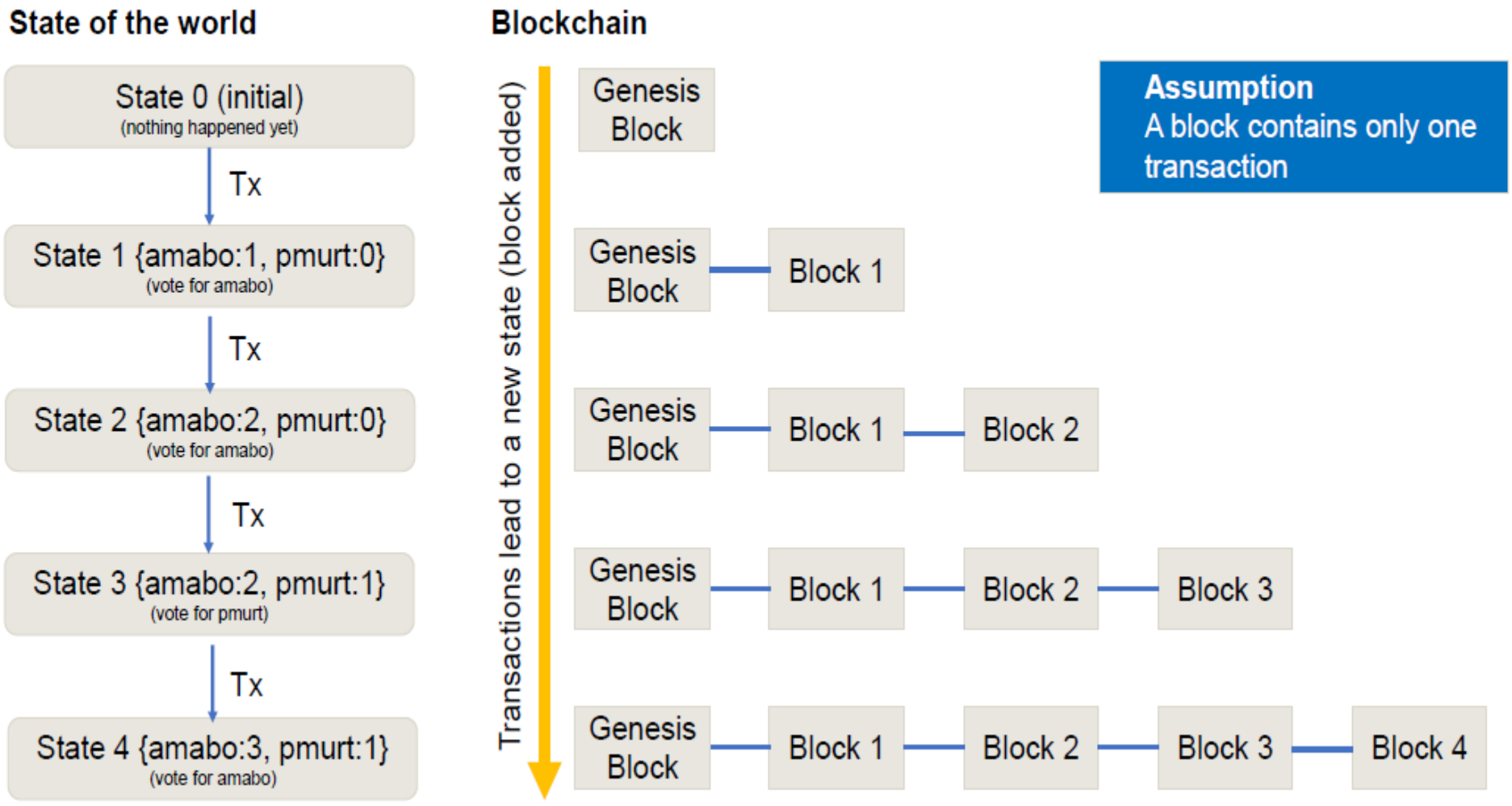
State 1 {amabo:1, pmurt:0}
(vote for amabo)

State 2 {amabo:2, pmurt:0}
(vote for amabo)

State 3 {amabo:2, pmurt:1}
(vote for pmurt)

State 4 {amabo:3, pmurt:1}
(vote for amabo)

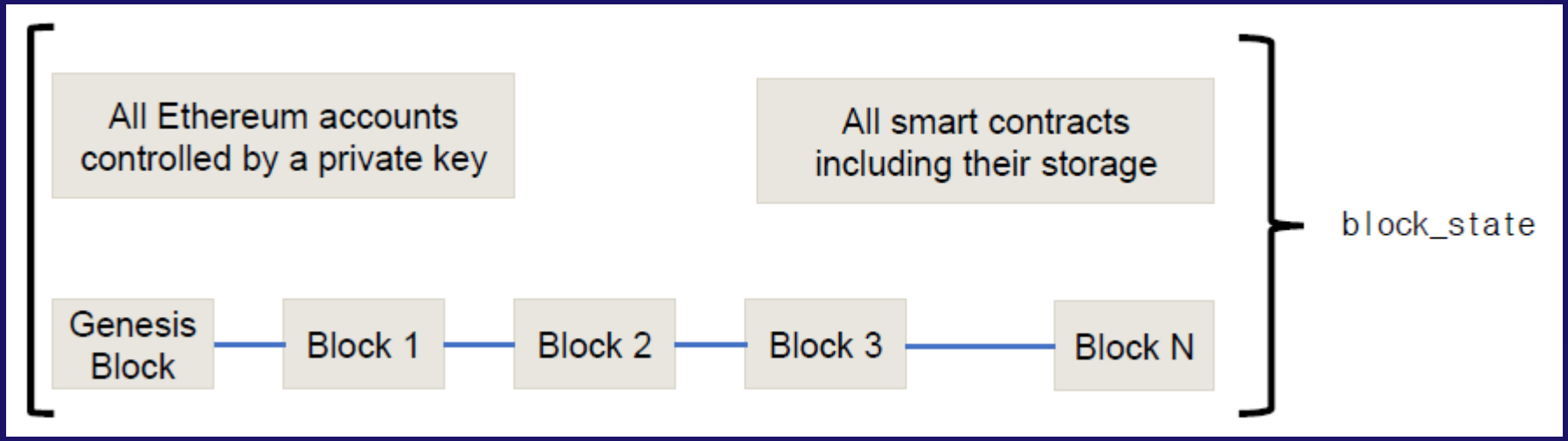
The blockchain as a state machine



Ethereum

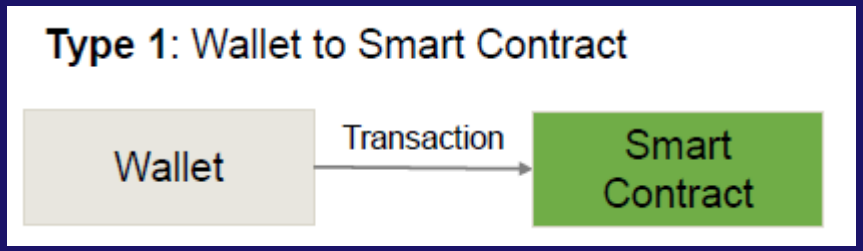
- The EVM specifies an execution model for state changes of the blockchain.
- Formally, the EVM can be specified by the following tuple:

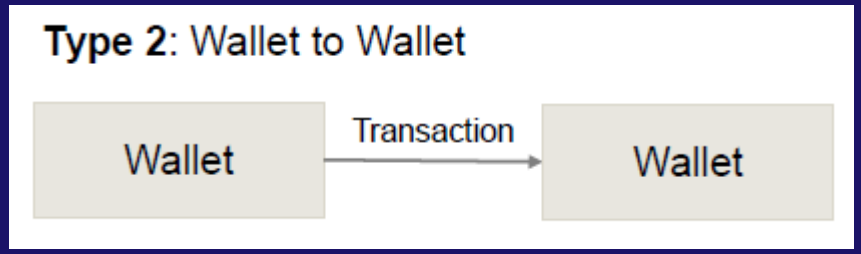
  (block_state, transaction, message, code, memory, stack, pc, gas)
- *32* byte instruction word size
- The block_state represents the global state of the whole blockchain including all accounts, contracts and storage



Ethereum

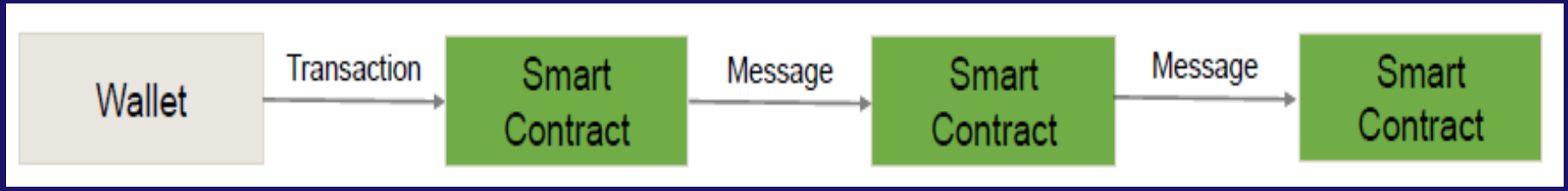
- A transaction is a signed data package that is always sent by a wallet and contains the following data field :
  - *Nonce : The transaction number for this account, starting with 0.*
  - *To : The address of the recipient of the message*
  - *Value : The amount of wei (ether) to transfer from the sender to the recipient*
  - *Gas Price : The price in wei that this transaction will pay for Gas*
  - *Gas Limit : The maximum amount of gas that this transaction can consume.*
  - *Data : Data that the code in the EVM processes*
  - *Signature (v r, s) : A signature identifying the sender*
- There are two types of transactions: From wallet to wallet and from wallet to smart contract



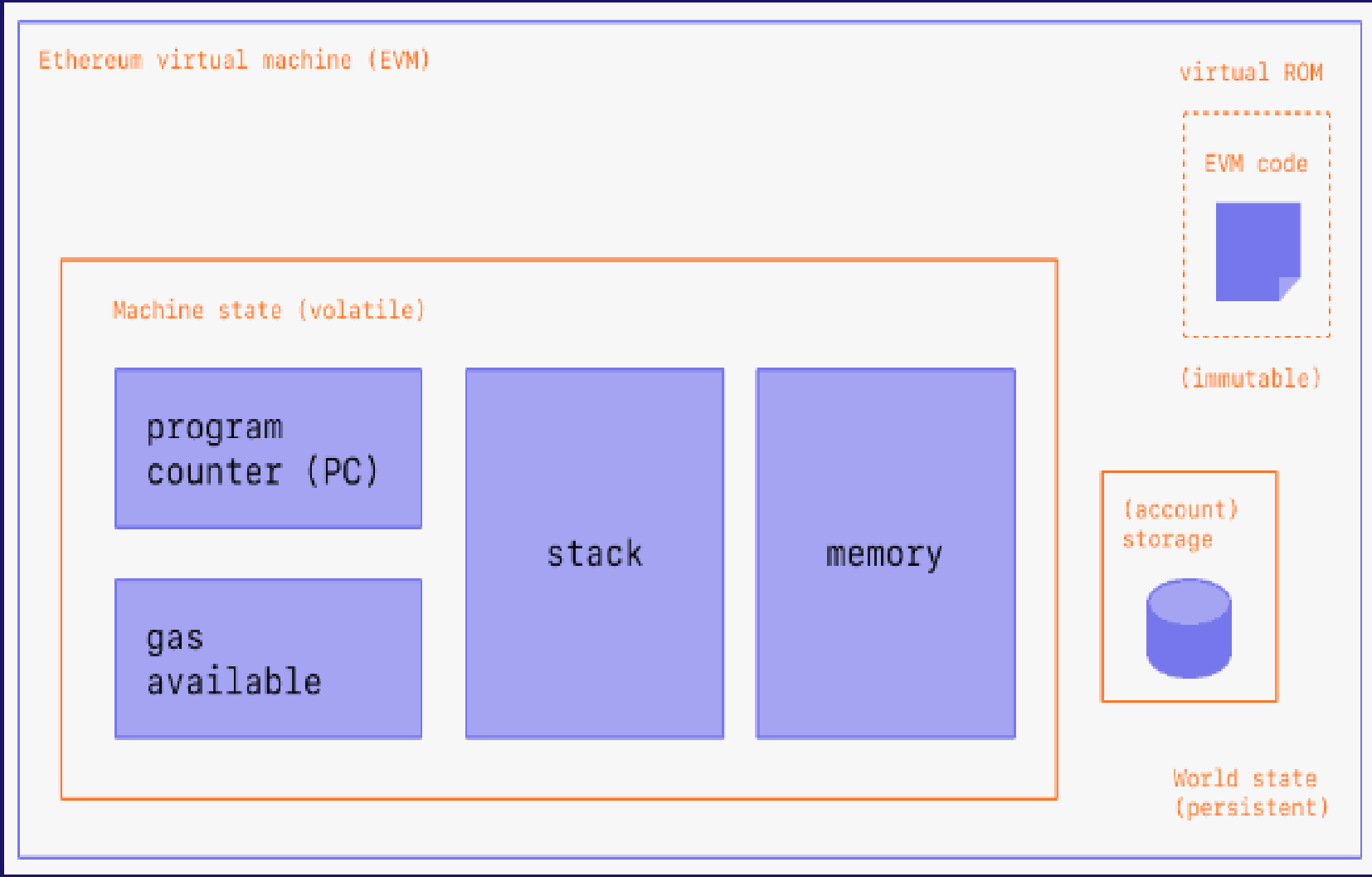


Ethereum

- A message is very similar to a transaction. Messages are only sent by contracts and exist only virtually, i.e. they are not mined into a block like transactions. It contains :
  - *The sender of the message (implicit)*
  - *The recipient of the message*
  - *The amount of ether to transfer alongside the message*
  - *An optional data field*
  - *A Gas limit value*
- Whenever a contract calls a method on another contract, a virtual messageis sent. Whenever a wallet calls a method on a contract, a transactionis sent.



Ethereum : EVM



Ethereum : EVM

- The EVM makes the code portable across different machines.
- The EVM is a stack-based processor.
- The EVM has 140 opcodes which allow it to be Turing complete.
- Each opcode takes 1 byte of storage space.
- The EVM also uses a 32 byte (256 bit) register stack which holds 1024 items.
- It also has a contract memory (non-persistent) for complicated operations.
- Reading from storage is free, but writing to storage is extremely expensive.

Ethereum : EVM

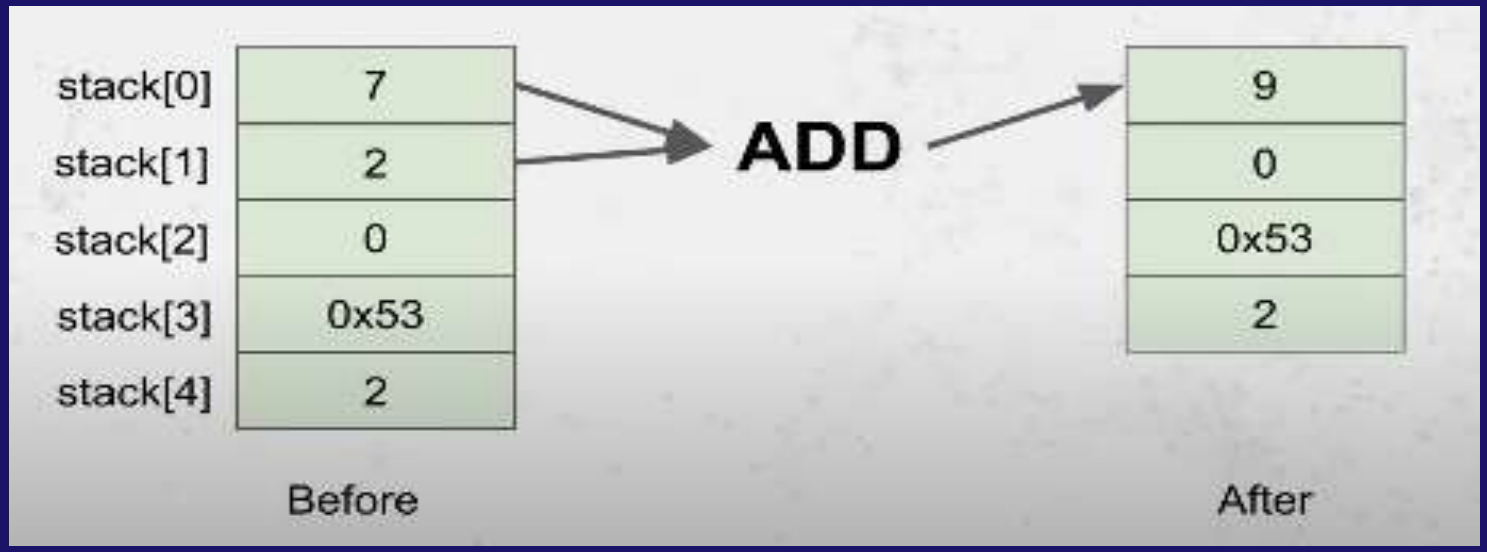
- Code : The code basically represents a smart contract as bytecode. For the EVM, a smart contract is a sequence of opcodes similar to assembly code.
- *Example :*

```
PUSH1 0x60
PUSH1 0x40
MSTORE
PUSH1 0x04
CALLDATASIZE
LT
PUSH2 0x00b6
JUMPI
PUSH4 0xffffffff
```

- Memory : An infinitely expandable byte array that is non-persistent and used as temporal storage during execution.

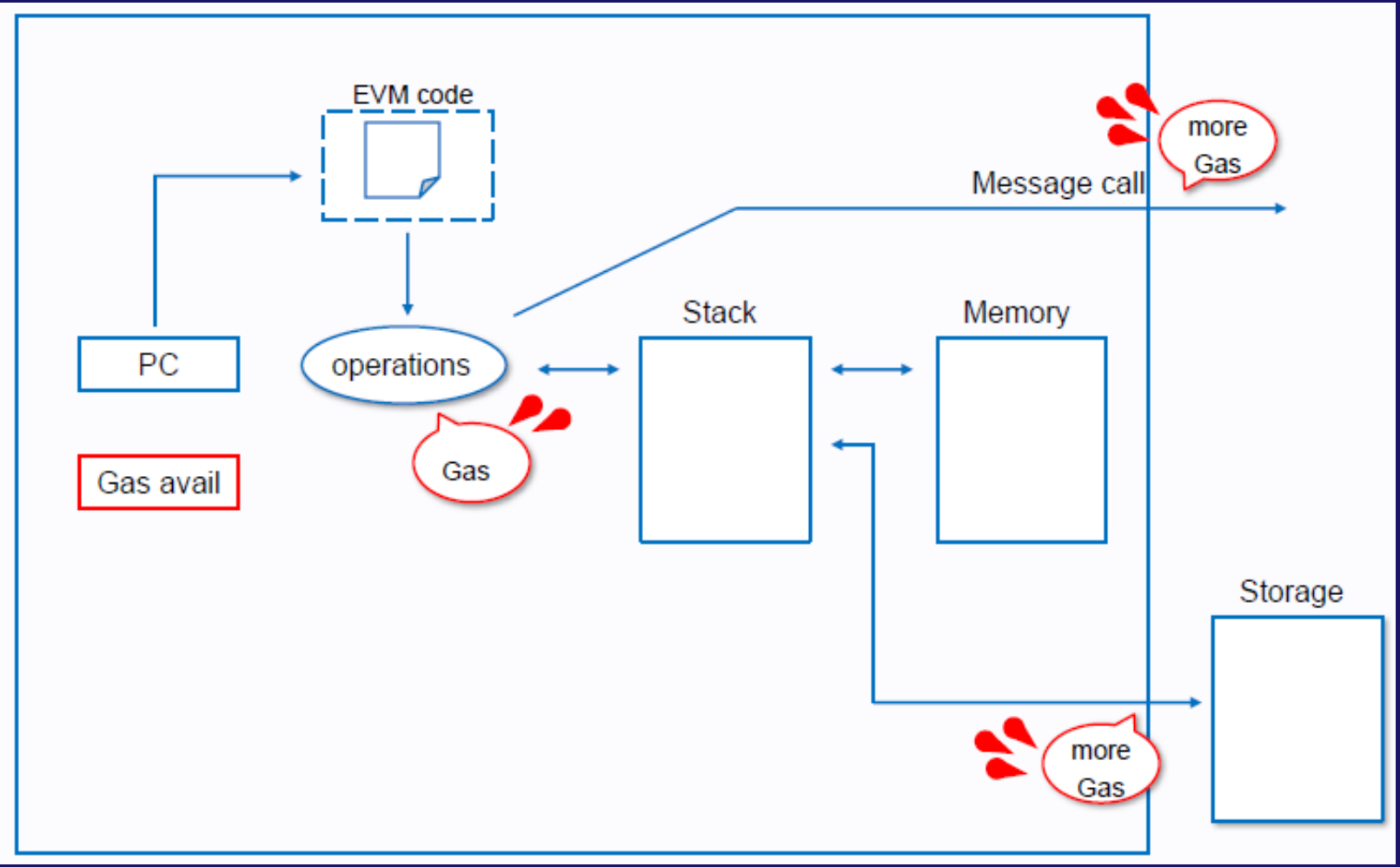
Ethereum : EVM

- The stack is also used as a fast, non-persistent buffer to which 32 byte (256 bit) values can be pushed and popped during execution.



- The program counter PC is always initialized with 0 and points to the position of the current opcode instruction.

Ethereum : Gas



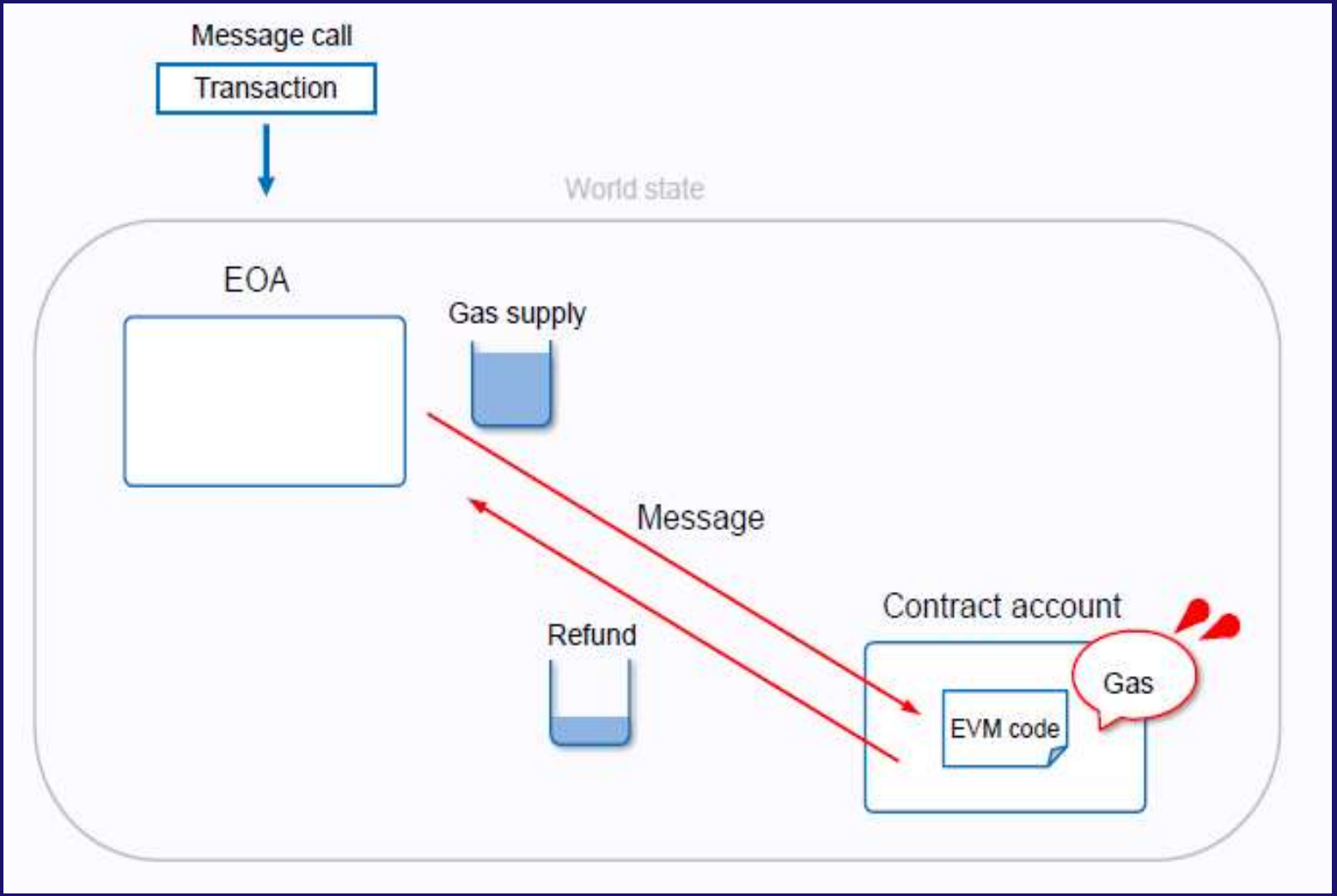
Ethereum : Gas

- Every executed opcode instruction uses a miner's computational resources and therefore costs a certain fee (called gas).
- Each opcode uses a certain amount of gas which may depend on the arguments of the operation, e.g., number of bytes to be allocated.
- At the beginning of a transaction the sender must specify a maximum amount of gas that he is willing to pay for the transaction to be executed.
- The sender can set an arbitrary amount of Ether he is willing to pay for each instruction called gas price.
- The final costs for each transaction are used *gas* * gas price.
- If a transaction requires more gas as the maximum specified gas, the transaction will fail. On the other hand, if it takes less, the sender only pays the gas that was used.

Ethereum : Gas

Opcode	Name	Description	Gas
`0x00`	STOP	Halts execution	0
`0x01`	ADD	Addition operation	3
`0x02`	MUL	Multiplication operation	5
`0x03`	SUB	Subtraction operation	3
`0x04`	DIV	Integer division operation	5
`0x05`	SDIV	Signed integer division operation (truncated)	5
`0x06`	MOD	Modulo remainder operation	5
`0x07`	SMOD	Signed modulo remainder operation	5
`0x08`	ADDMOD	Modulo addition operation	8
`0x09`	MULMOD	Modulo multiplication operation	8
`0x0a`	EXP	Exponential operation	10*
`0x0b`	SIGNEXTEND	Extend length of two's complement signed integer	5
`0x0c` - `0x0f`	Unused	Unused	
`0x10`	LT	Less-than comparison	3
`0x11`	GT	Greater-than comparison	3
`0x12`	SLT	Signed less-than comparison	3
`0x13`	SGT	Signed greater-than comparison	3
`0x14`	EQ	Equality comparison	3
`0x15`	ISZERO	Simple not operator	3

Ethereum : Gas and Fee



Ethereum : Account types

- Ethereum supports two types of accounts identified by 32-byte address : *Externally Owned Account (EOA) and Smart Contract Account*
- Externally Owned Account
  - *Owned by some external entity (person, corporation, etc.)*
  - *Accounts that are controlled by a ECDSA private key do not have any code stored on the blockchain.*
  - *Can be seen as the default wallet of a user.*
  - *It can sign transactions, issue smart contract functions calls and send Ether from one account to another.*
  - *The origin of any transaction is always an account controlled and signed by the private key.*
- Smart Contract Account
  - *Account controlled by the contract code.*
  - *Contracts can send messages to other accounts, both externally and smart contracts.*
  - *Code execution triggered by transactions or function calls (msg)*

Ethereum : Account types

- On an abstract level, an Ethereum account is a 4-tuple containing the following data : (nonce, balance, contract_code, storage)

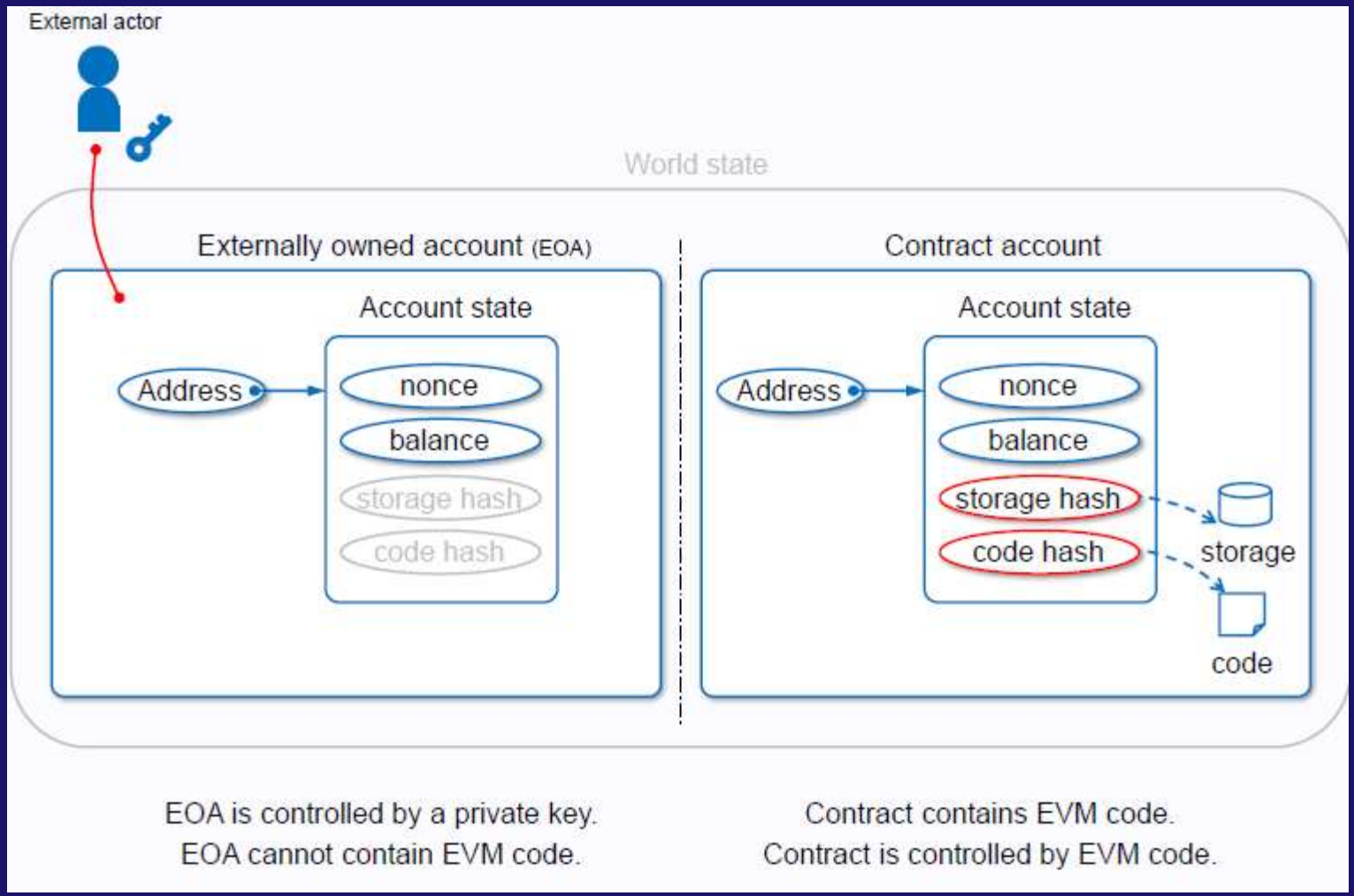
nonce
An increasing number that is attached to any transaction to prevent replay attacks and double spending.

balance
The current account balance of the account in Ether.

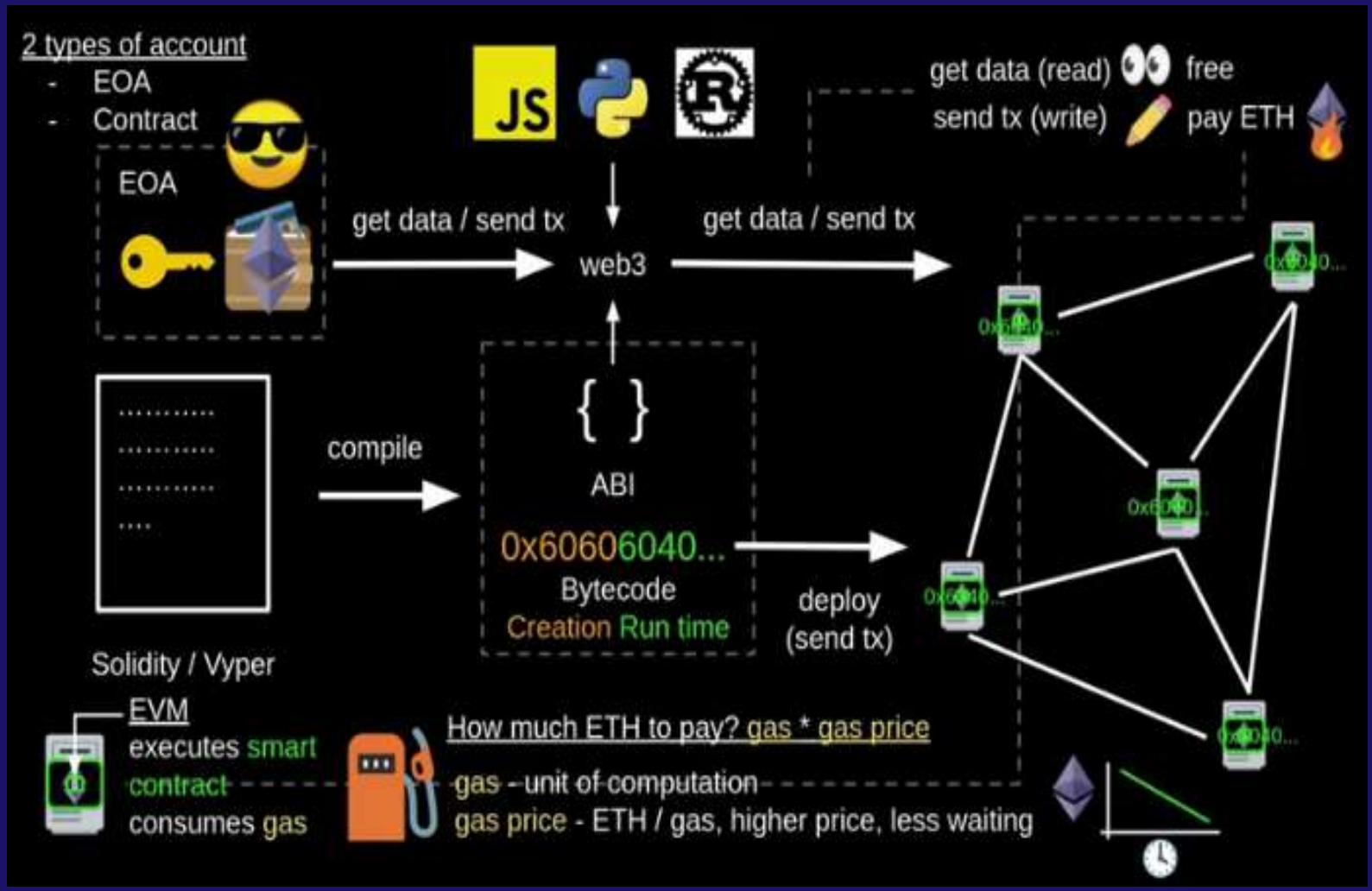
contract_code
The bytecode representation of the account. If no contract code is present, then the account is externally controlled.

storage
The data storage used by the account and empty by default. Only contract accounts can have their own storage.

Ethereum : Account types



Ethereum


2 types of account
- EOA
- Contract
EOA
JS
get data (read)
free
send tx (write)
pay ETH
get data / send tx
web3
get data / send tx
compile
ABI
0x60606040...
Bytecode
Creation Run time
deploy
(send tx)
Solidity / Vyper
EVM
executes smart
contract
consumes gas
How much ETH to pay? gas * gas price
gas - unit of computation
gas price - ETH / gas, higher price, less waiting