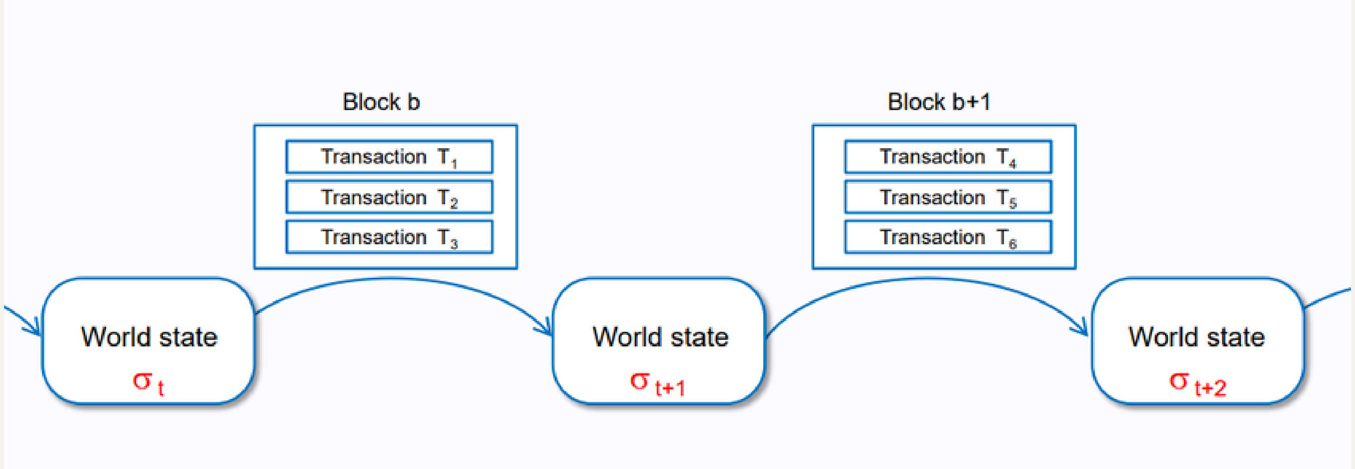
TRANSACTIONS AND EVM

Created: Sept 2022
Last Edited: Sept 2022

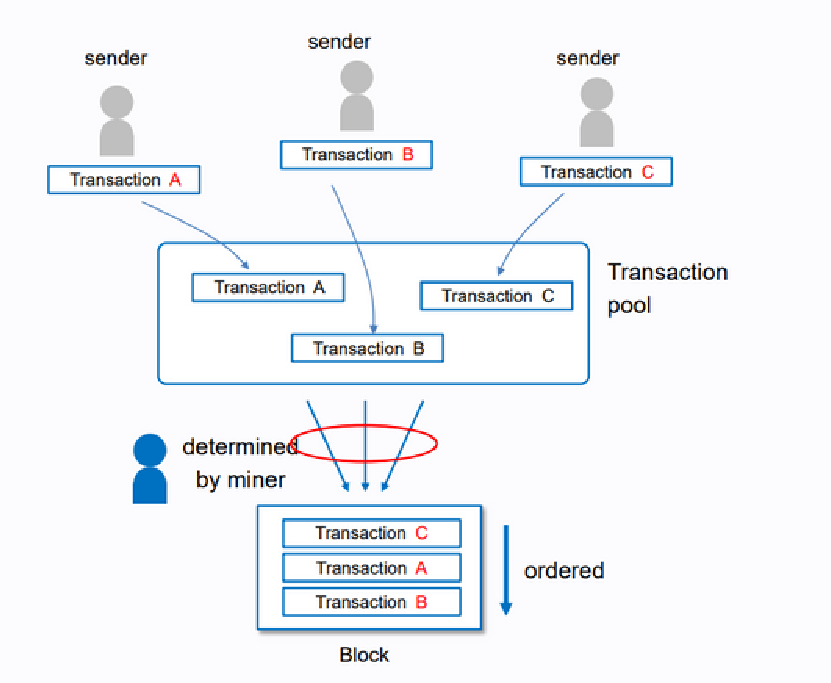
UofT: CSCD71F22
-David Liu, Founder of dApp Technology Inc.

TRANSACTIONS AND BLOCKS I



Each Transaction contains instructions to move the blockchain from one "World State" to another.

TRANSACTIONS AND BLOCKS II



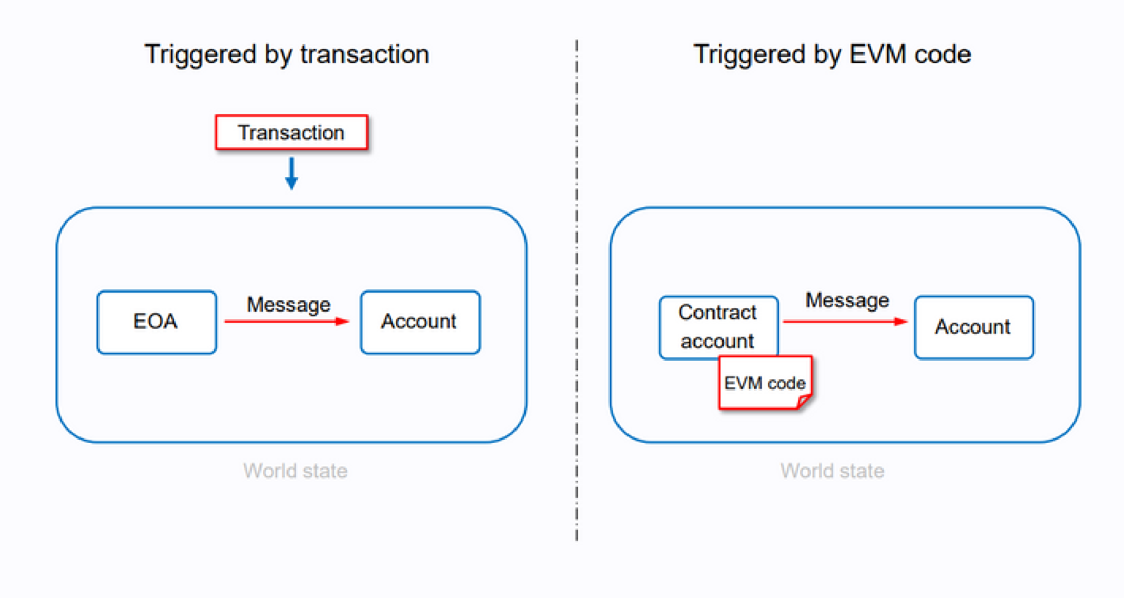
The transactions are selected from the Transaction pool (also called Mempool) to a block, in an order the Validators choose.

All transactions in a block share the same timestamp.

Definitions from Yellow Paper

Transaction: A piece of data, signed by an External Actor. It represents either a Message or a new Autonomous Object. Transactions are recorded into each block of the blockchain.

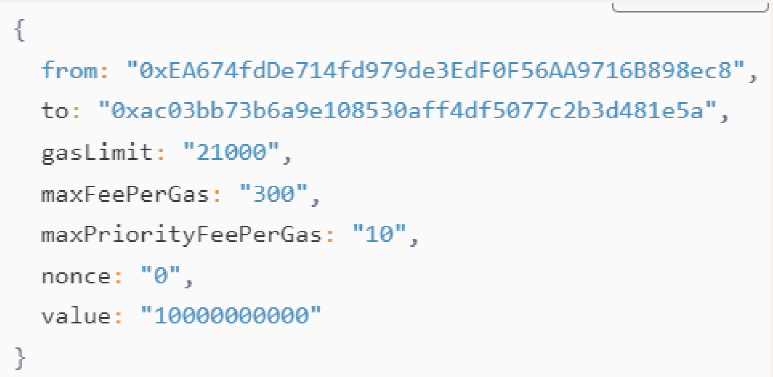
Message: Data (as a set of bytes) and Value (specified as Ether) that is passed between two Accounts, either through the deterministic operation of an Autonomous Object or the cryptographically secure signature of the Transaction



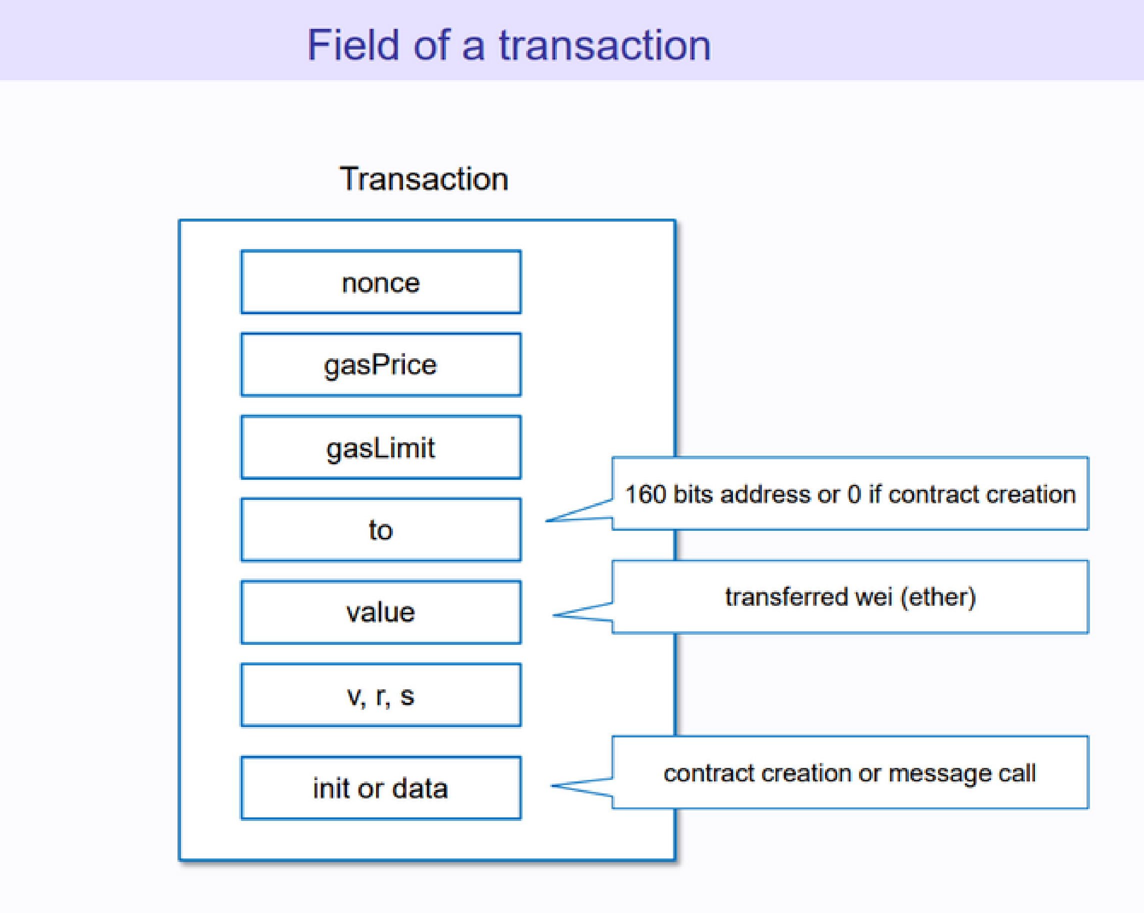
Transaction triggers an associated message.

EVM can also send a message.

Transaction Example



```
{
  from: "0xEA674fdDe714fd979de3EdF0F56AA9716B898ec8",
  to: "0xac03bb73b6a9e108530aff4df5077c2b3d481e5a",
  gasLimit: "21000",
  maxFeePerGas: "300",
  maxPriorityFeePerGas: "10",
  nonce: "0",
  value: "10000000000"
}
```



Field of a transaction

Transaction

- nonce
- gasPrice
- gasLimit
- to
- value
- v, r, s
- init or data

to: 160 bits address or 0 if contract creation

value: transferred wei (ether)

init or data: contract creation or message call

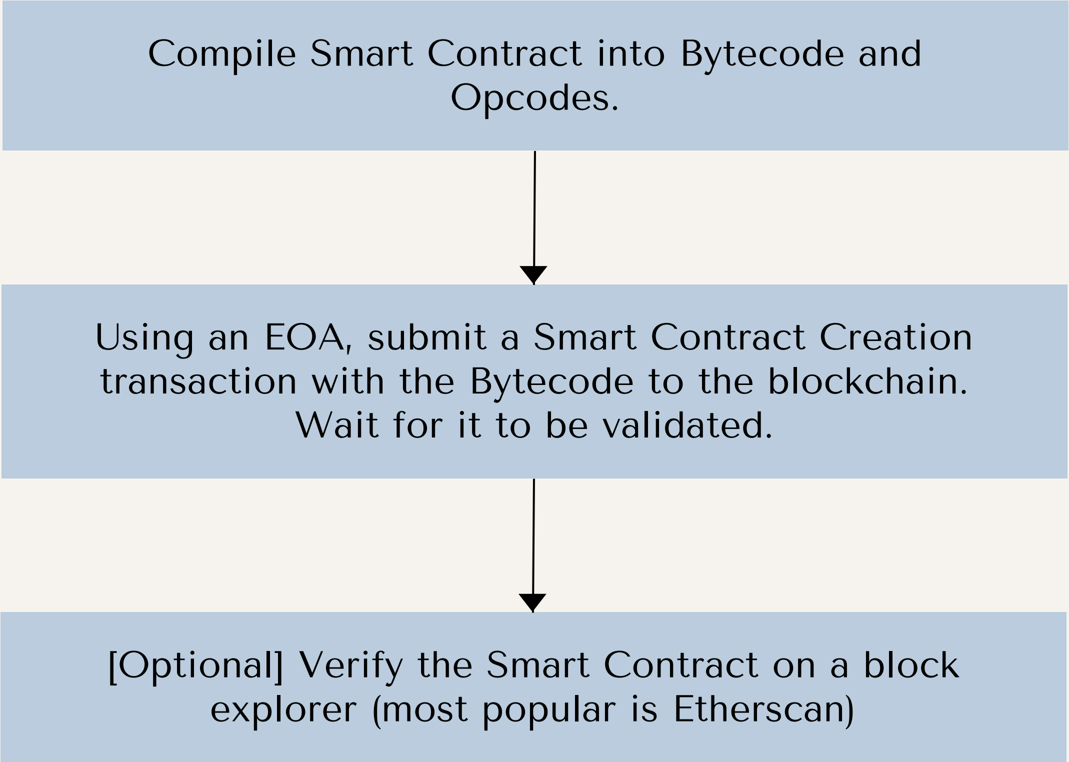
v, r, s are the values for the transaction's signature

WHAT ARE SMART CONTRACTS?

Smart Contracts are code that are stored and executed on the blockchain computer. In Ethereum, the computer is called the Ethereum Virtual Machine (EVM).

Smart Contracts can be executed as by anyone and anytime as determined by the predefined rule in the code.

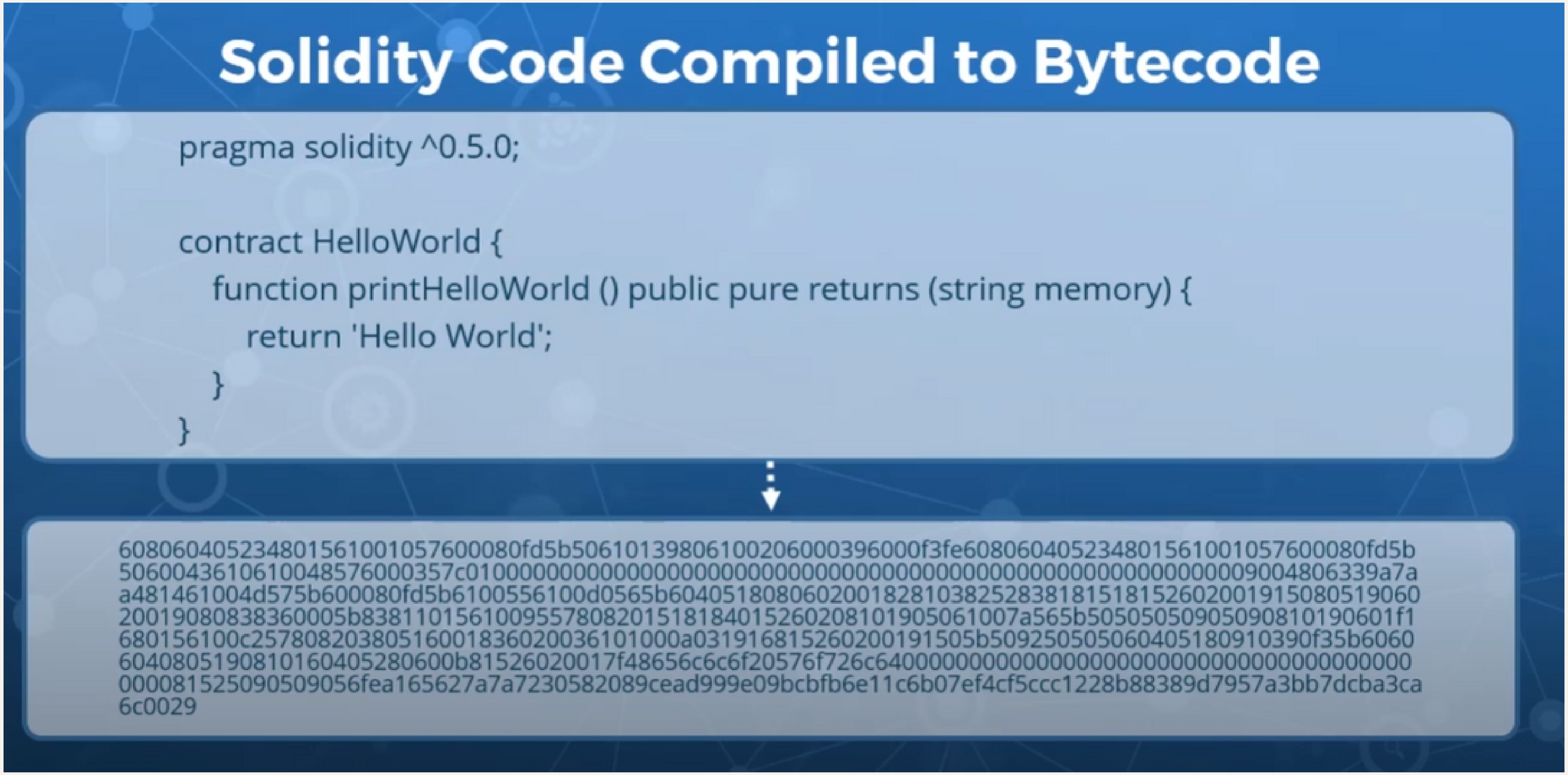
SMART CONTRACT DEPLOYMENT



Compile Smart Contract into Bytecode and Opcodes.

Using an EOA, submit a Smart Contract Creation transaction with the Bytecode to the blockchain. Wait for it to be validated.

[Optional] Verify the Smart Contract on a block explorer (most popular is Etherscan)

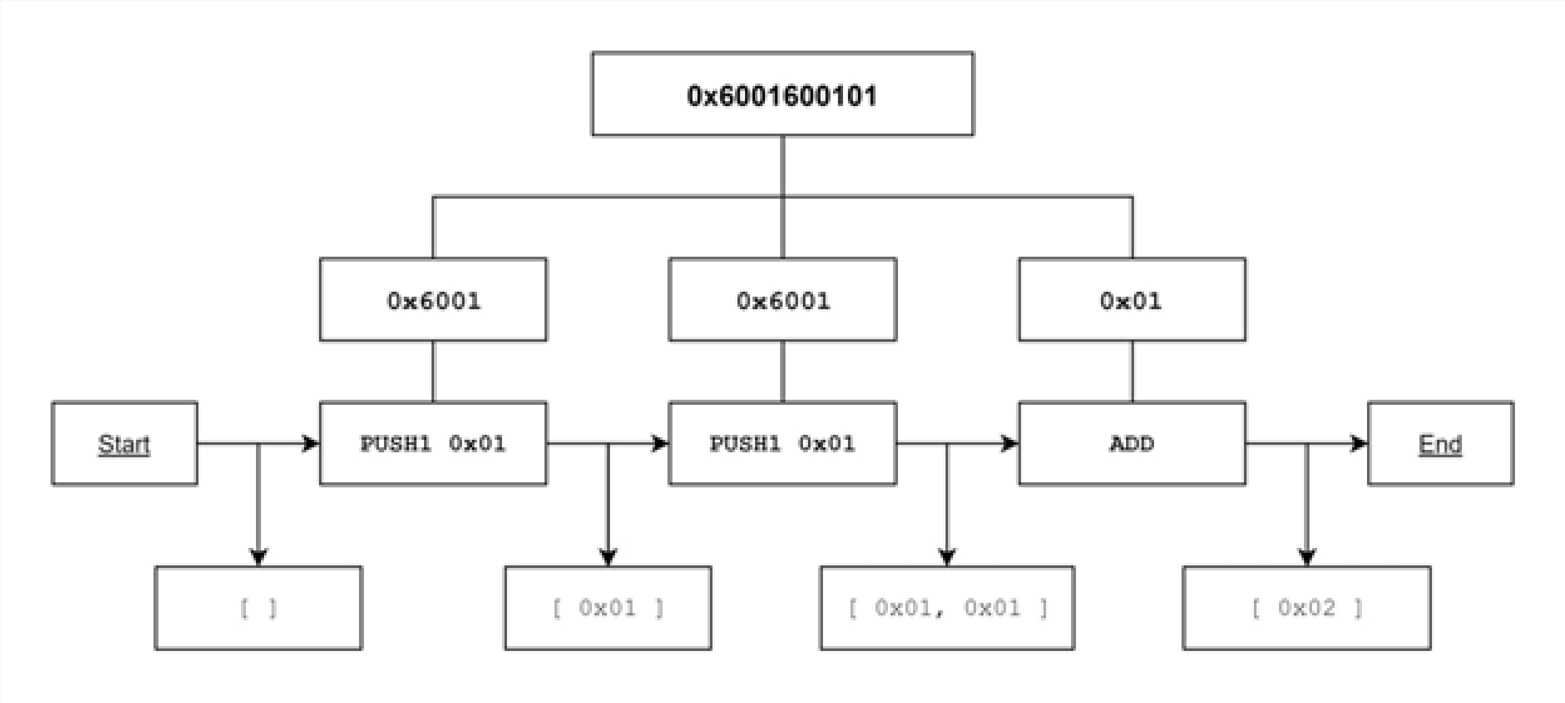


Solidity Code Compiled to Bytecode

```
pragma solidity ^0.5.0;

contract HelloWorld {
   function printHelloWorld () public pure returns (string memory) {
      return 'Hello World';
   }
}
```

```
608060405234801561001057600080fd5b50610139806100206000396000f3fe608060405234801561001057600080fd5b
5060043610610048576000357c0100000000000000000000000000000000000000000000000000000000900480633 9a7a
a481461004d575b600080fd5b6100556100d0565b60405180806020018281038252838181518152602001915080519060
200190808383600005b838110156100955780820151818401526020810190506100 7a565b50505050905090810190601f1
680156100c25780820380516001836020036101000a031916815260200191505b509250505060405180910390f35b6060
6040805190810160405280600b81526020017f48656c6c6f20576f726c6400000000000000000000000000000000000000
000081525090509056fea165627a7a7230582089cead999e09bcbfb6e11c6b07ef4cf5ccc1228b88389d7957a3bb7dcba3ca
6c0029
```

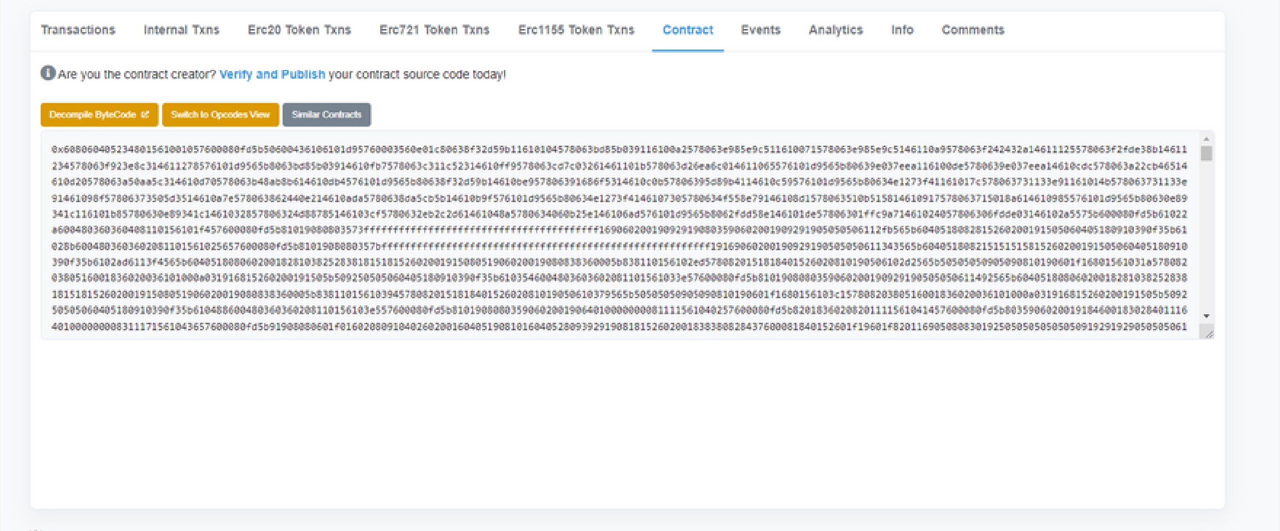
PARSING BYTECODE INTO OPCODES



VERIFYING SMART CONTRACT I

Verifying is done by sending the data used in the Contract Creation Transaction and the source code of the smart contract to the Block Explorer. The Block Explorer will then hash the data to check if it is the same hash as the deployed smart contract.

Unverified Smart Contract on Etherscan

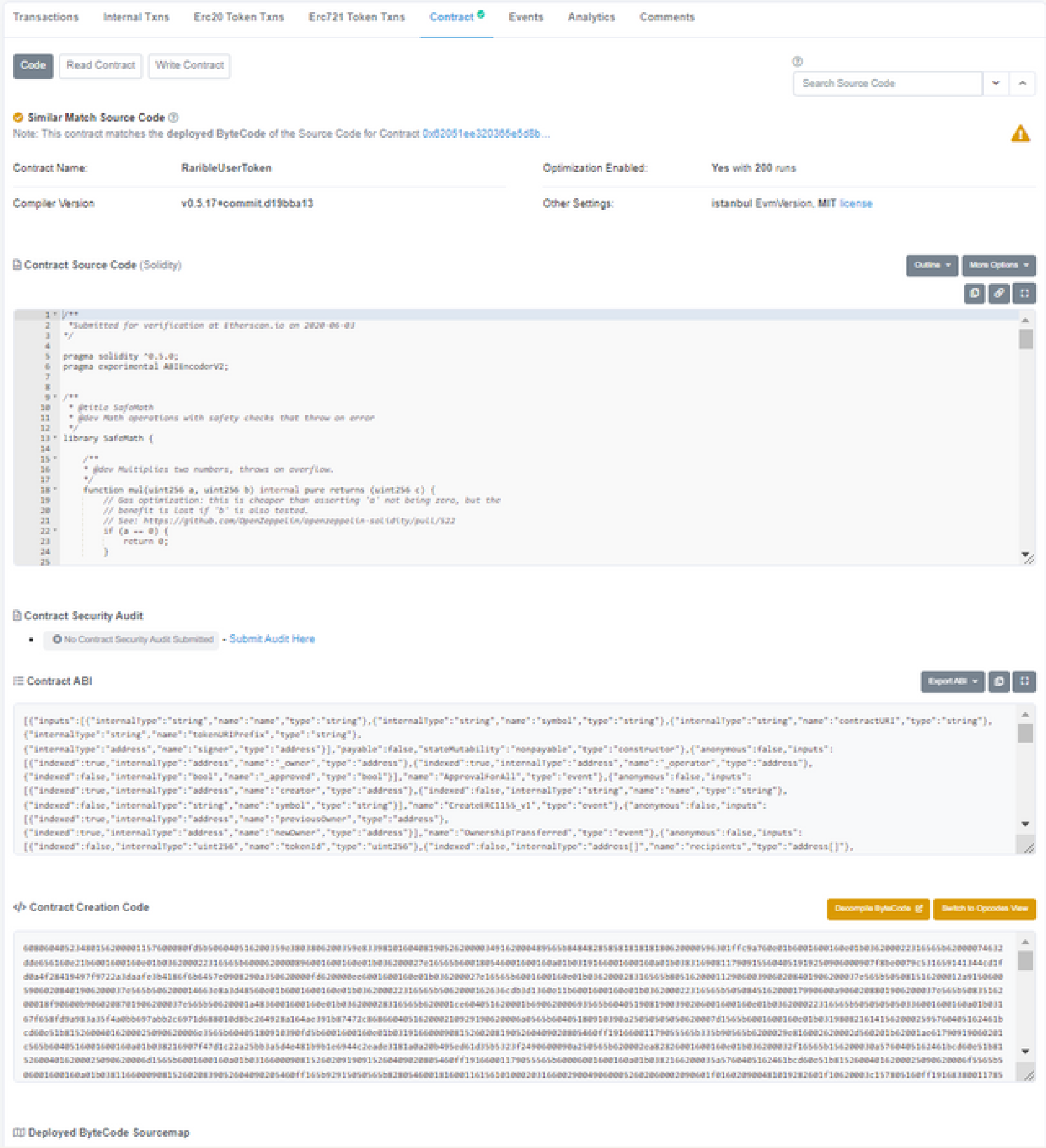


Are you the contract creator? Verify and Publish your contract source code today!

```
0x608060405234801561001057600080fd5b50600436106101d95760003560e01c80638f32d59b11610104578063bd85b039116100a2578063e985e9c511610071578063e985e9c5146110a9578063f242432a14611125578063f2fde38b14611
234578063f923e8c314611278576101d9565b8063bd85b03914610fb7578063c311c52314610ff9578063cd7c03261461101b578063d26ea6c014611065576101d9565b80639e037eea116100de5780639e037eea14610cdc578063a22cb46514
610d20578063a50aa5c314610d70578063b48ab8b614610db4576101d9565b80638f32d59b14610be957806391686f5314610c0b57806395d89b4114610c59576101d9565b80634e1273f41161017c578063731133e91161014b578063731133e
91461098f57806373505d3514610a7e578063862440e214610ada5780638da5cb5b14610b9f576101d9565b80634e1273f4146107305780634f558e79146108d1578063510b51581461091757806371 5018a614610985576101d9565b80630e89
341c116101b85780630e89341c1461032857806324d88785146103cf5780632eb2c2d61461048a5780634060b25e146106ad576101d9565b8062fdd58e146101de57806301ffc9a71461024057806306fdde03146102a5575b600080fd5b61022
a600480360360408110156101f457600080fd5b81019080803573ffffffffffffffffffffffffffffffffffffffff16906020019092919080359060200190929190505050611 2fb565b6040518082815260200191505060405180910390f35b61
028b600480360360208110156102565760008 0fd5b8101908080357bffffffffffffffffffffffffffffffffffffffffffffffffffffffff191690602001909291905050506113 43565b6040518082151515158152602001915050604051809 10
390f35b6102ad6113f4565b6040518080602001828103825283818151815260200191508051906020019080838360005b838110156102ed578082015181840152602081019050 6102d2565b50505050905090810190601f16801561031a578082
0380516001836020036101000a031916815260200191505b509250505060405180910390f35b610354600480360360208110156103 3e57600080fd5b810190808035906020019092919050505061149 2565b60405180806020018281038252838
181518152602001915080519060200190808383600 05b83811015610394578082015181840152602081019050610379565b50505050905090810190601f1680156103c157808203805160018360200 36101000a031916815260200191505b5092
50505060405180910390f35b6104886004803603602081101561 03e557600080fd5b810190808035906020019064010000000081111561040257600080fd5b82018360208201111561041457600080fd5b8035906020019184600183028401116
401000000008311171561043657600080fd5b91908080601f016020809104026020016040519081016040528093929190818152602001838380828437600081840152601f19601f82011690508083019250505050505050919291929050505061
```

VERIFYING SMART CONTRACT II

Verified Smart Contract on Etherscan



Similar Match Source Code

Note: This contract matches the deployed ByteCode of the Source Code for Contract 0x82051ee320365e5d8b...

Contract Name:	RaribleUserToken	Optimization Enabled:	Yes with 200 runs
Compiler Version	v0.5.17+commit.d19bba13	Other Settings:	istanbul EvmVersion, MIT license

Contract Source Code (Solidity)

```
/**
 *Submitted for verification at Etherscan.io on 2020-06-03
*/

pragma solidity ^0.5.0;
pragma experimental ABIEncoderV2;


/**
 * @title SafeMath
 * @dev Math operations with safety checks that throw on error
 */
library SafeMath {

    /**
    * @dev Multiplies two numbers, throws on overflow.
    */
    function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
        // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }
```

Contract Security Audit

- No Contract Security Audit Submitted - Submit Audit Here

Contract ABI

[{"inputs":[{"internalType":"string","name":"name","type":"string"},{"internalType":"string","name":"symbol","type":"string"},{"internalType":"string","name":"contractURI","type":"string"},
{"internalType":"string","name":"tokenURIPrefix","type":"string"},
{"internalType":"address","name":"signer","type":"address"}],"payable":false,"stateMutability":"nonpayable","type":"constructor"},{"anonymous":false,"inputs":
[{"indexed":true,"internalType":"address","name":"_owner","type":"address"},{"indexed":true,"internalType":"address","name":"_operator","type":"address"},
{"indexed":false,"internalType":"bool","name":"_approved","type":"bool"}],"name":"ApprovalForAll","type":"event"},{"anonymous":false,"inputs":
[{"indexed":true,"internalType":"address","name":"creator","type":"address"},{"indexed":false,"internalType":"string","name":"name","type":"string"},
{"indexed":false,"internalType":"string","name":"symbol","type":"string"}],"name":"CreateERC1155_v1","type":"event"},{"anonymous":false,"inputs":
[{"indexed":true,"internalType":"address","name":"previousOwner","type":"address"},
{"indexed":true,"internalType":"address","name":"newOwner","type":"address"}],"name":"OwnershipTransferred","type":"event"},{"anonymous":false,"inputs":
[{"indexed":false,"internalType":"uint256","name":"tokenId","type":"uint256"},{"indexed":false,"internalType":"address[]","name":"recipients","type":"address[]"},

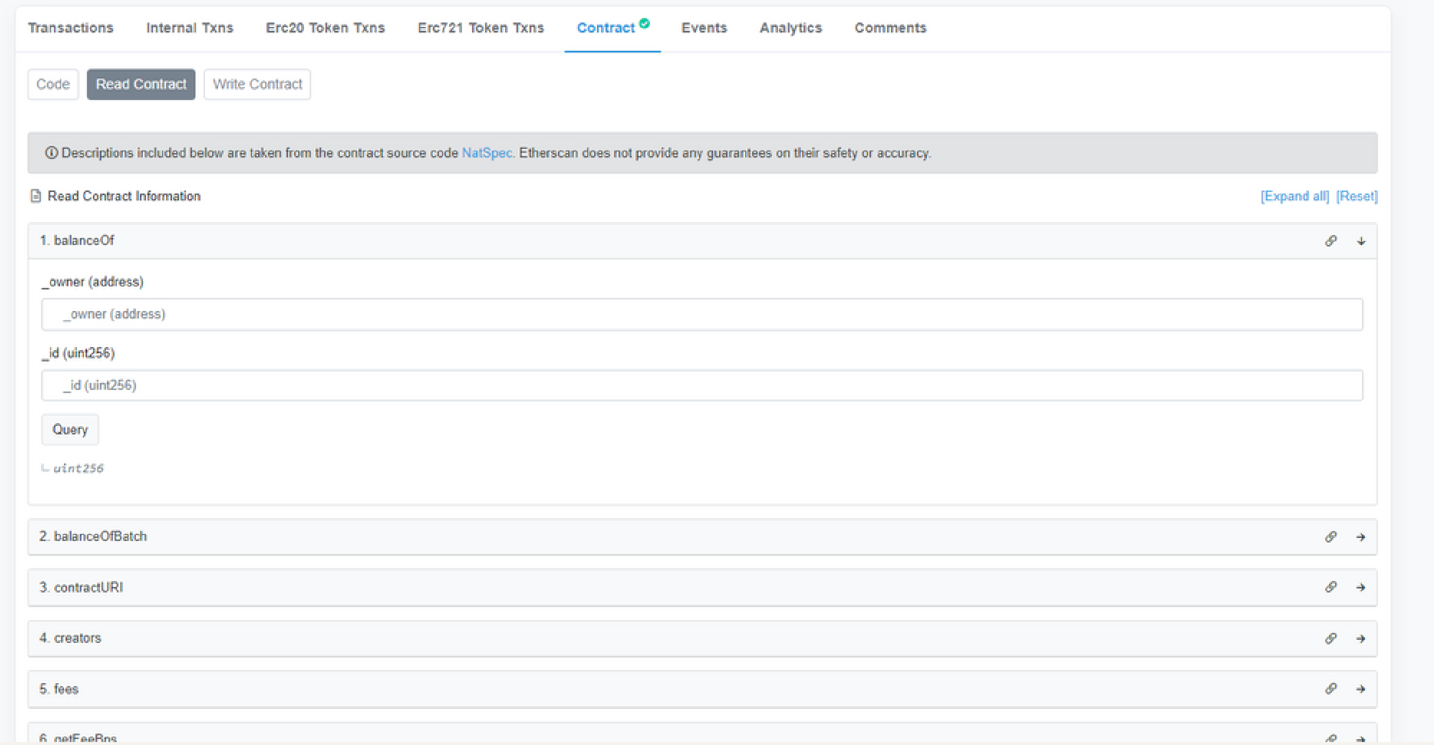
Contract Creation Code

[illegible]

Deployed ByteCode Sourcemap

VERIFYING SMART CONTRACT III

Reading from a verified Smart Contract on Etherscan



ⓘ Descriptions included below are taken from the contract source code NatSpec. Etherscan does not provide any guarantees on their safety or accuracy.

Read Contract Information

1. balanceOf

_owner (address)

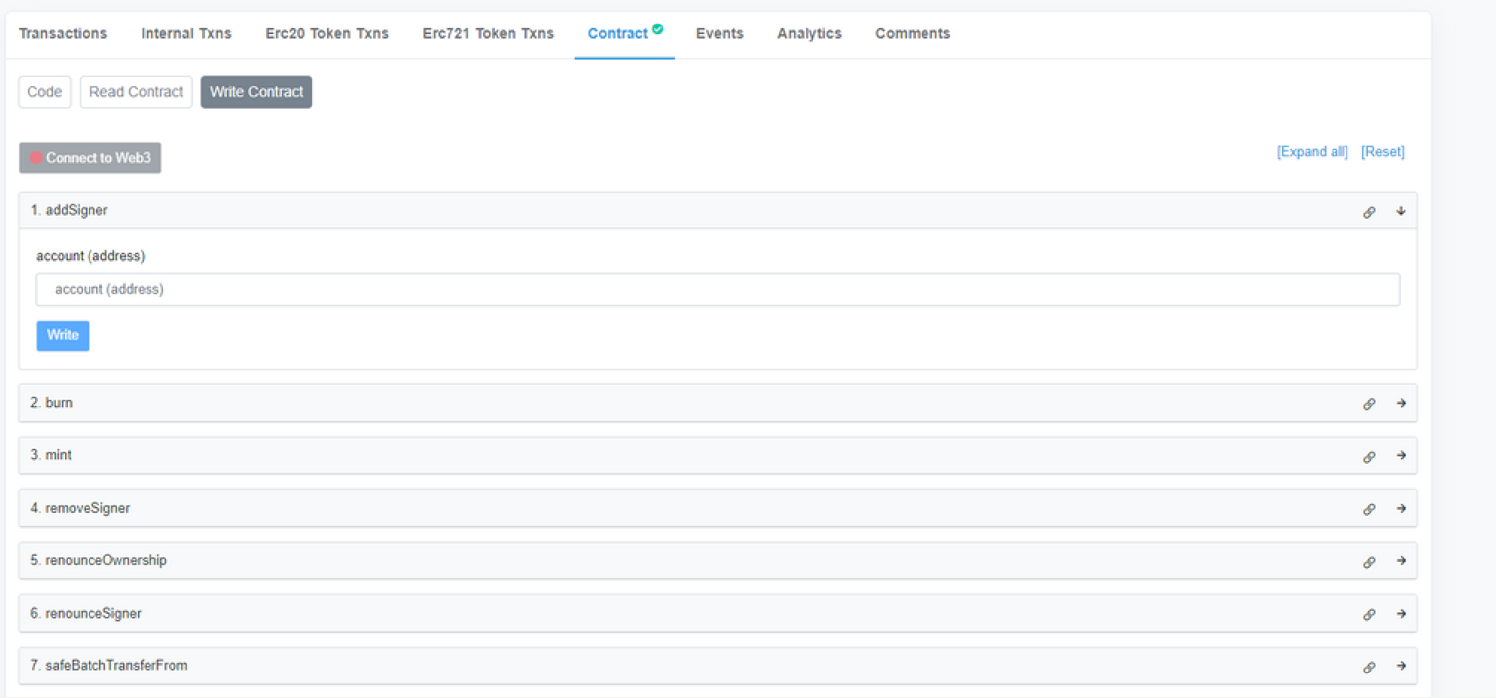
_id (uint256)

uint256

2. balanceOfBatch
3. contractURI
4. creators
5. fees
6. getFeeBps

VERIFYING SMART CONTRACT IV

Writing to a verified Smart Contract on Etherscan



1. addSigner

account (address)

account (address)

2. burn

3. mint

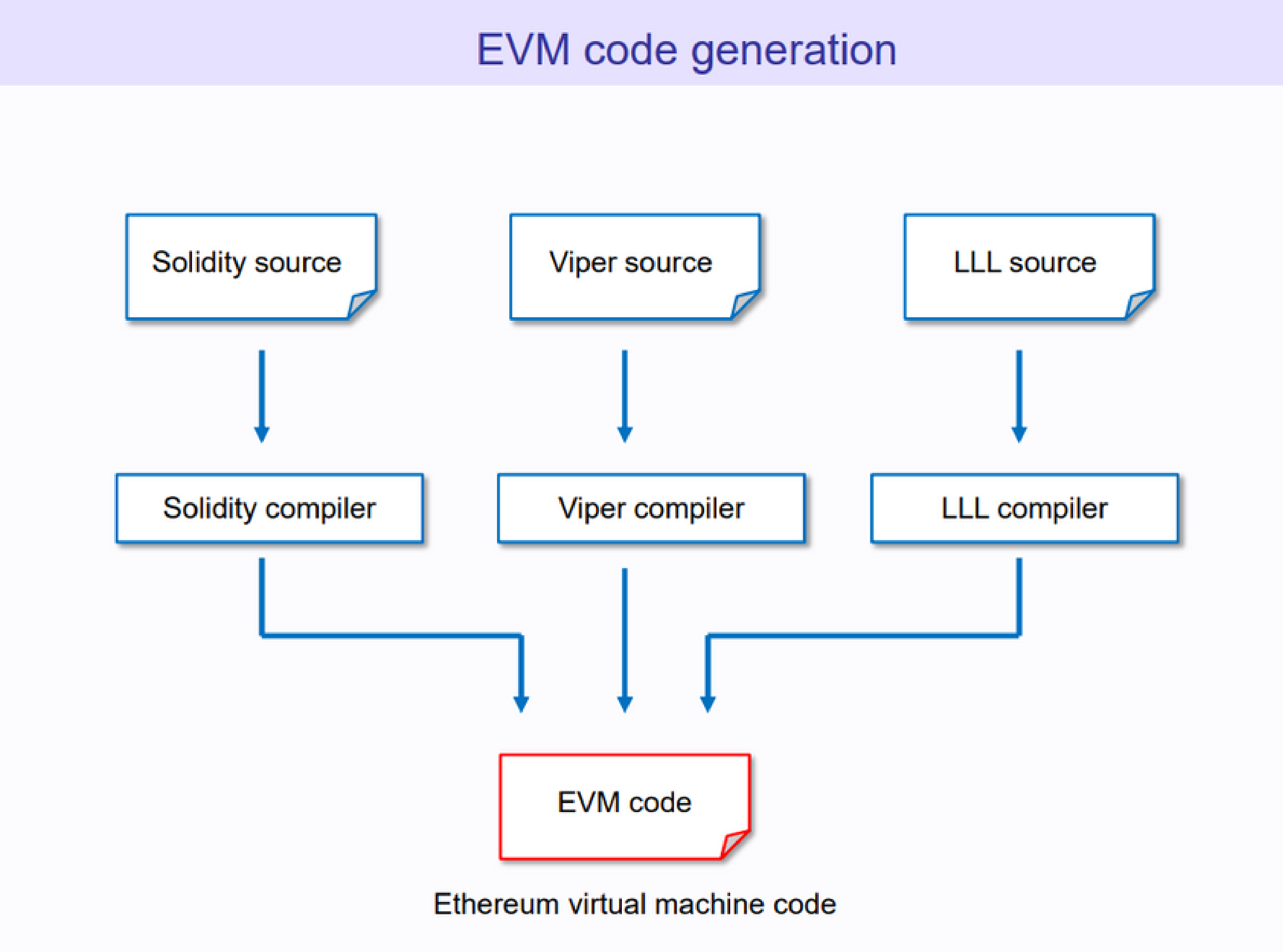
4. removeSigner

5. renounceOwnership

6. renounceSigner

7. safeBatchTransferFrom

EVM I



EVM code generation

Solidity source

Viper source

LLL source

Solidity compiler

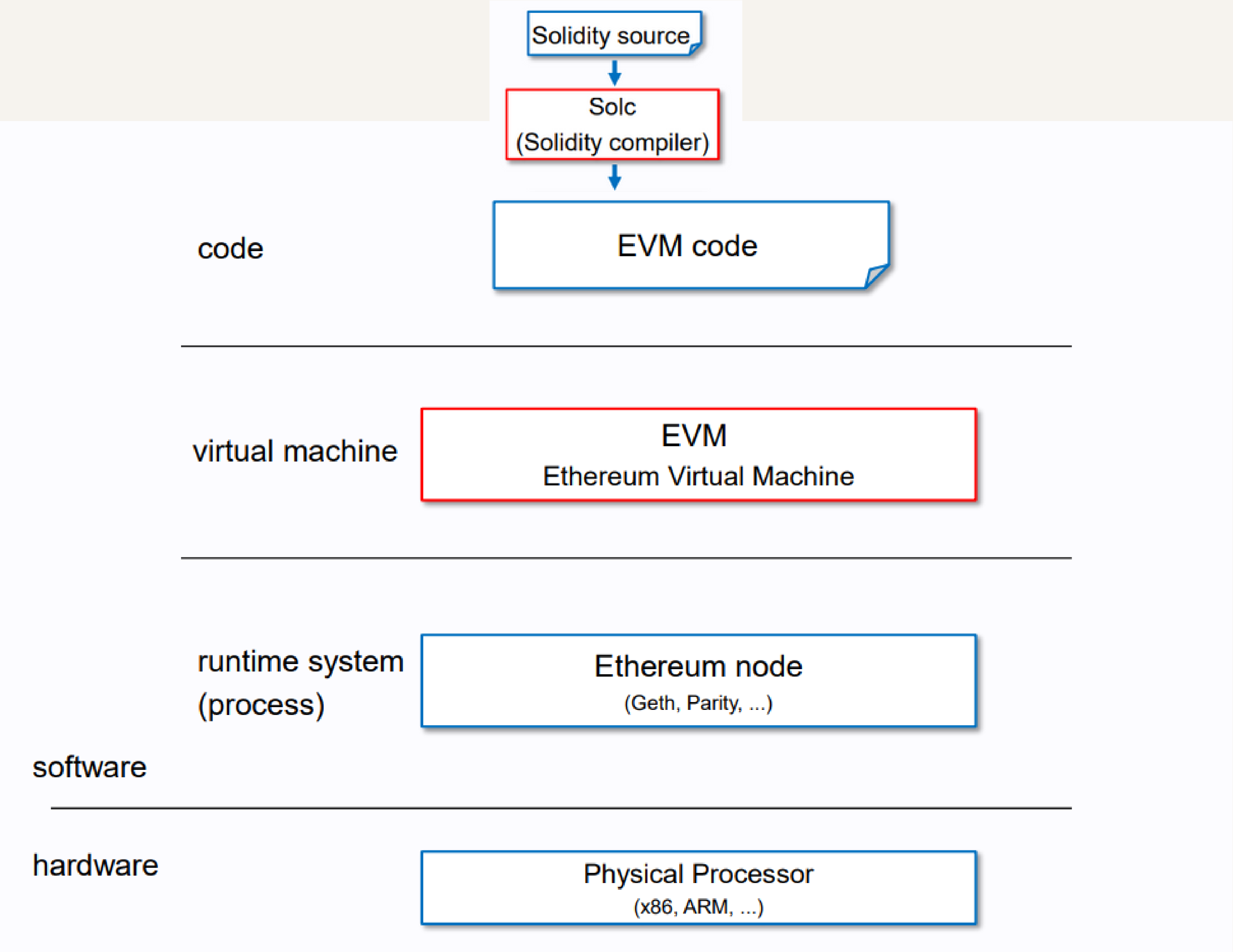
Viper compiler

LLL compiler

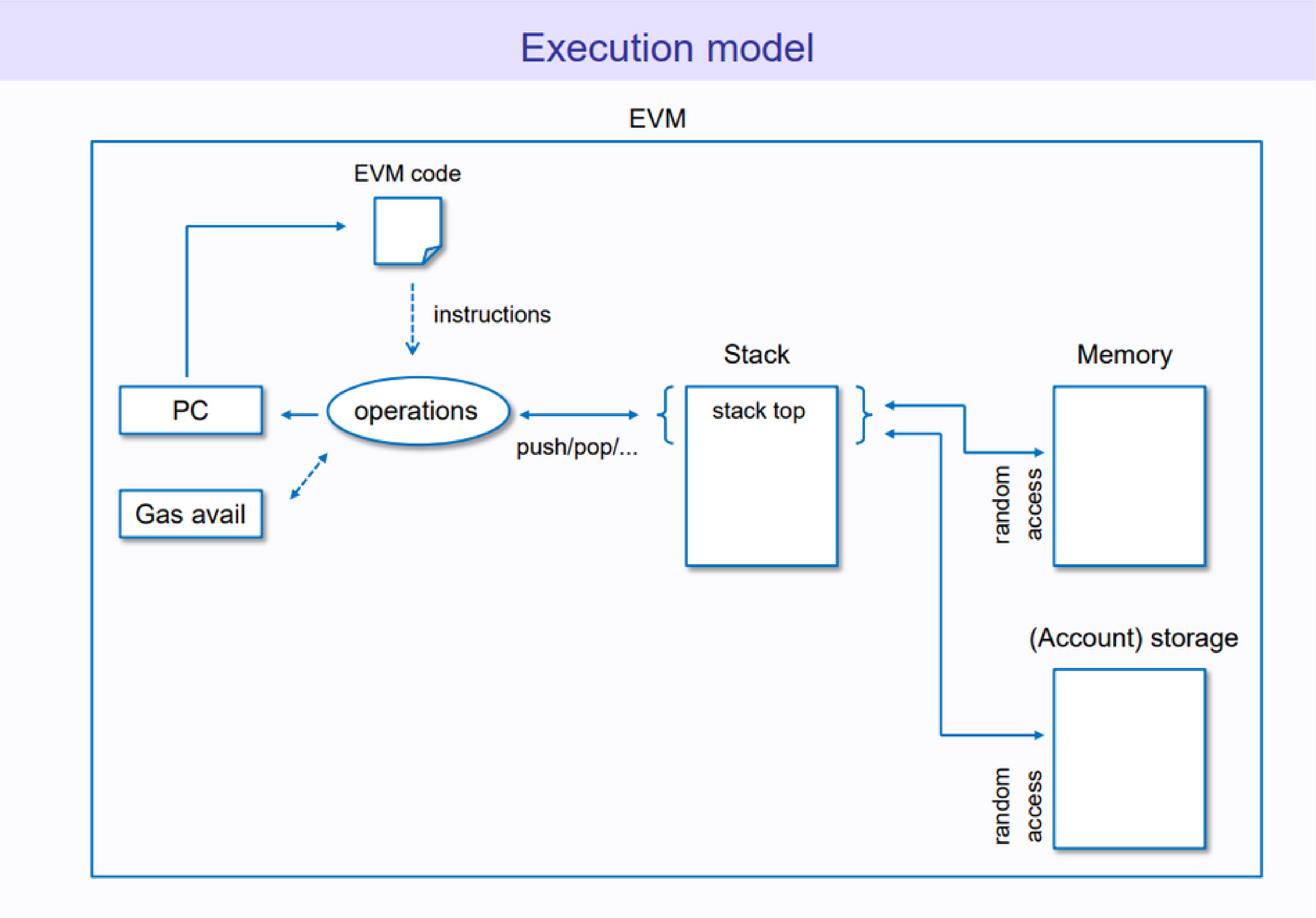
EVM code

Ethereum virtual machine code

EVM II

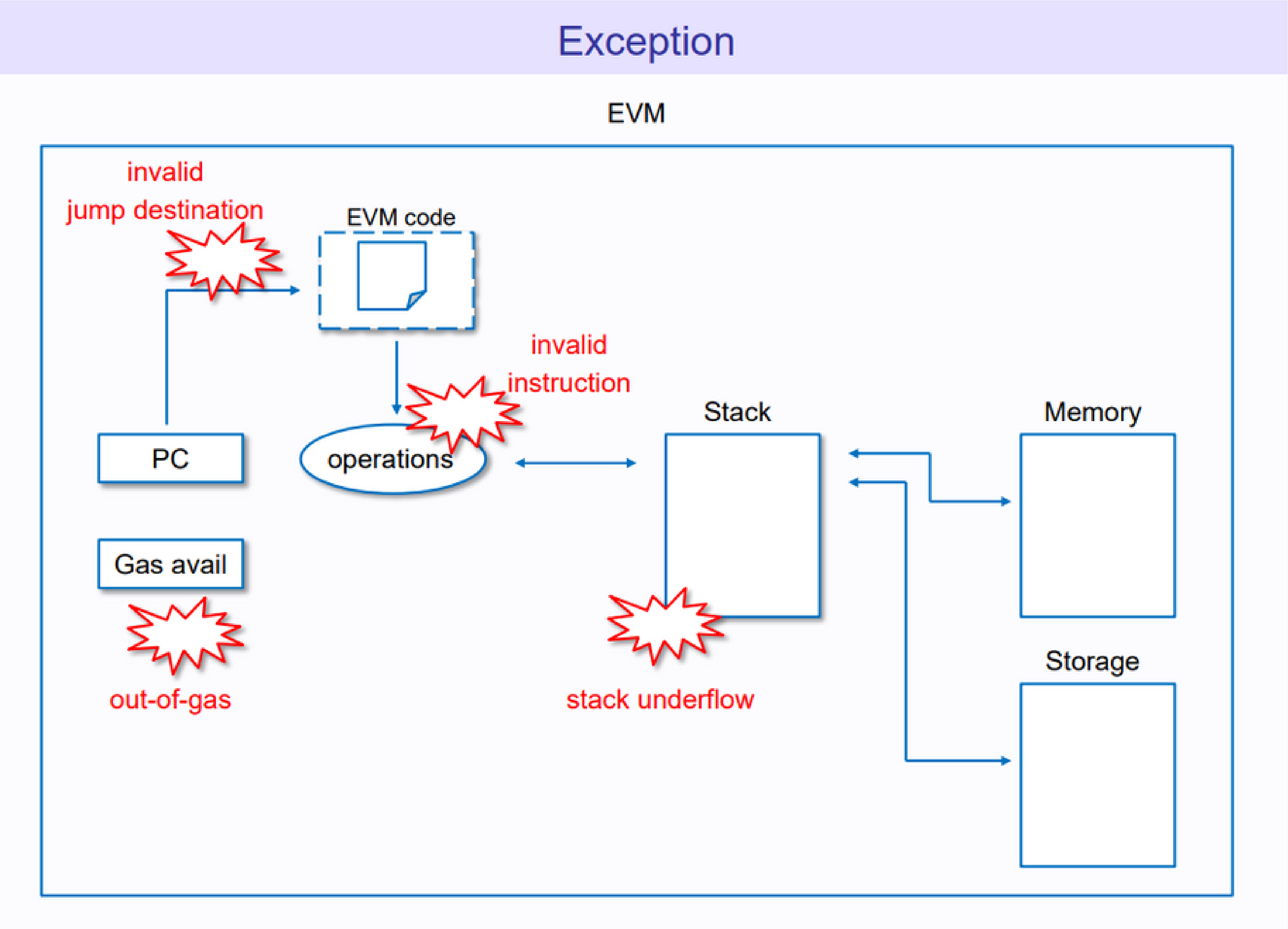

Solidity source
Solc
(Solidity compiler)
code
EVM code
virtual machine
EVM
Ethereum Virtual Machine
runtime system
(process)
Ethereum node
(Geth, Parity, ...)
software
hardware
Physical Processor
(x86, ARM, ...)

EVM III



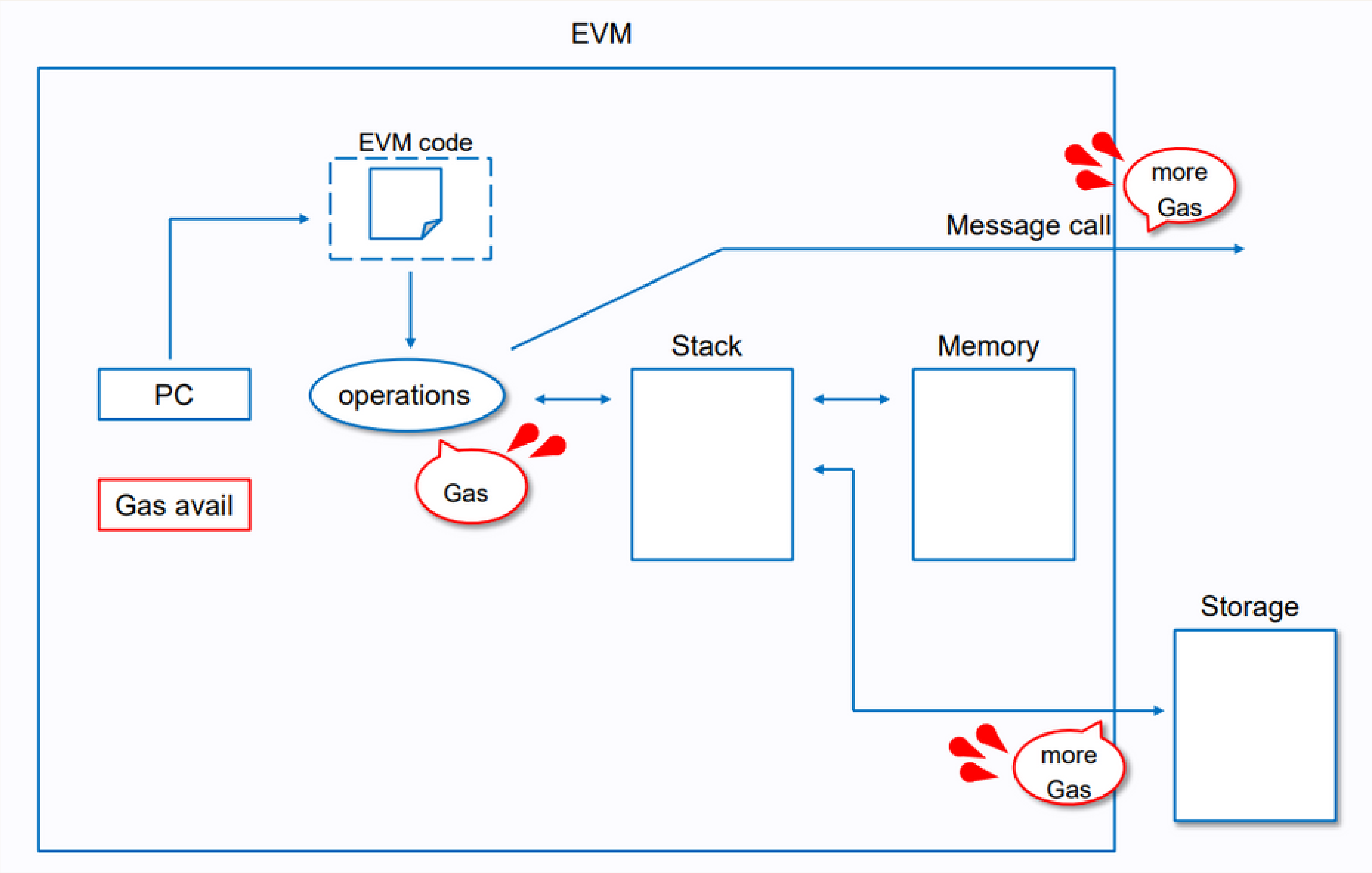
Execution model

EVM IV

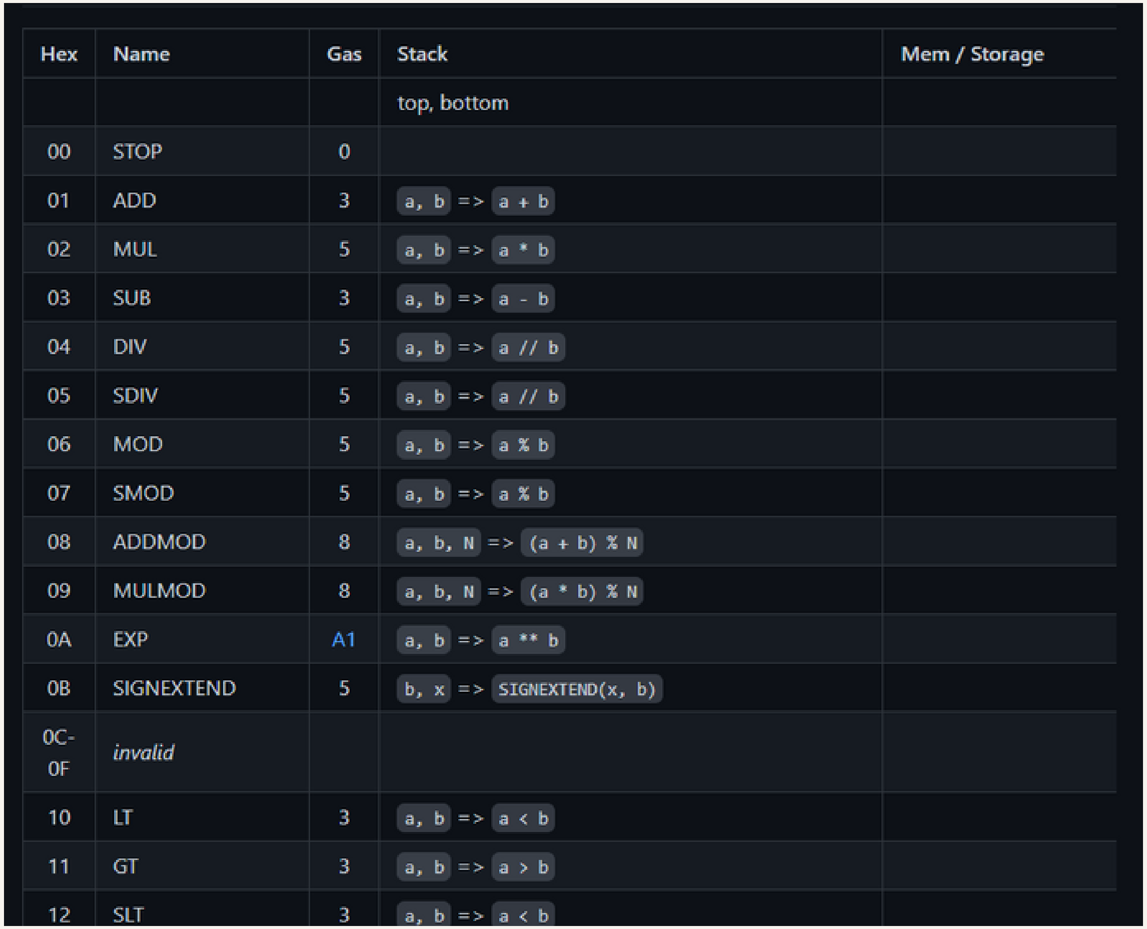


Exception

EVM V

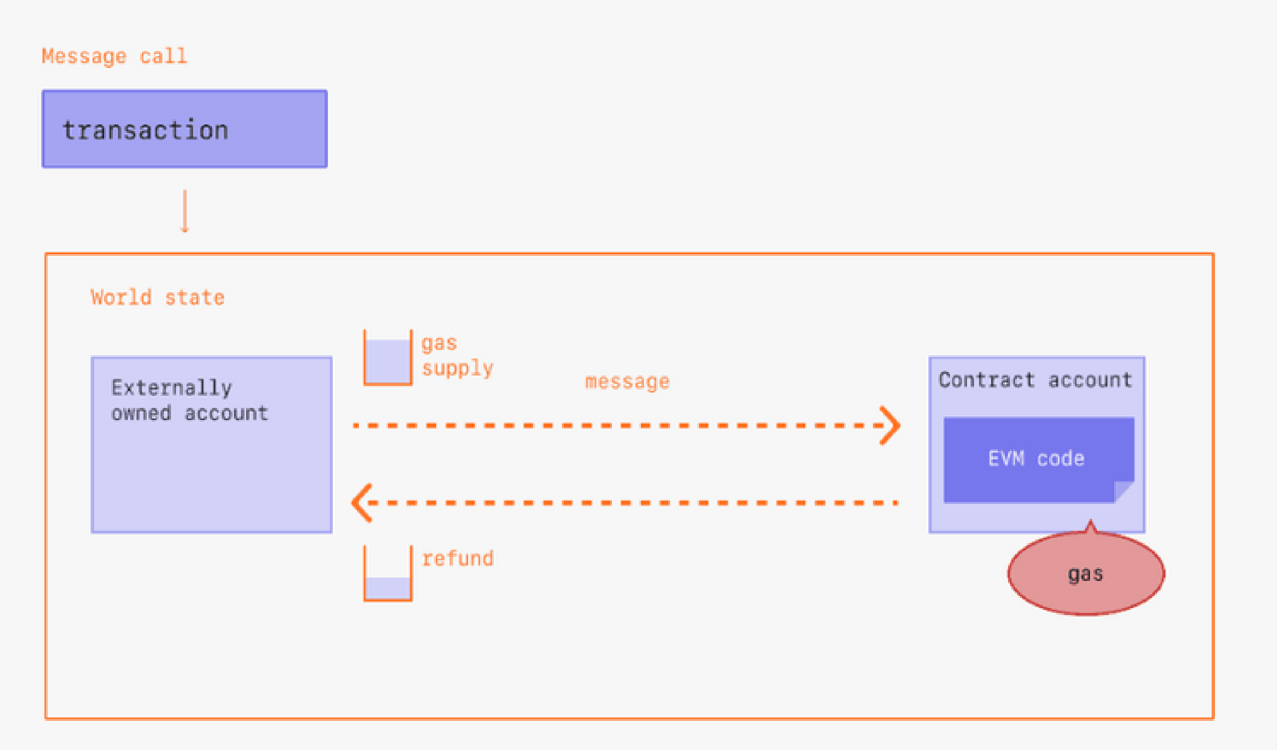

EVM
EVM code
more Gas
Message call
PC
operations
Stack
Memory
Gas
Gas avail
Storage
more Gas

GAS COSTS I



Hex	Name	Gas	Stack	Mem / Storage
			top, bottom	
00	STOP	0		
01	ADD	3	`a, b` => `a + b`	
02	MUL	5	`a, b` => `a * b`	
03	SUB	3	`a, b` => `a - b`	
04	DIV	5	`a, b` => `a // b`	
05	SDIV	5	`a, b` => `a // b`	
06	MOD	5	`a, b` => `a % b`	
07	SMOD	5	`a, b` => `a % b`	
08	ADDMOD	8	`a, b, N` => `(a + b) % N`	
09	MULMOD	8	`a, b, N` => `(a * b) % N`	
0A	EXP	A1	`a, b` => `a ** b`	
0B	SIGNEXTEND	5	`b, x` => `SIGNEXTEND(x, b)`	
0C-0F	*invalid*			
10	LT	3	`a, b` => `a < b`	
11	GT	3	`a, b` => `a > b`	
12	SLT	3	`a, b` => `a < b`	

GAS COSTS II

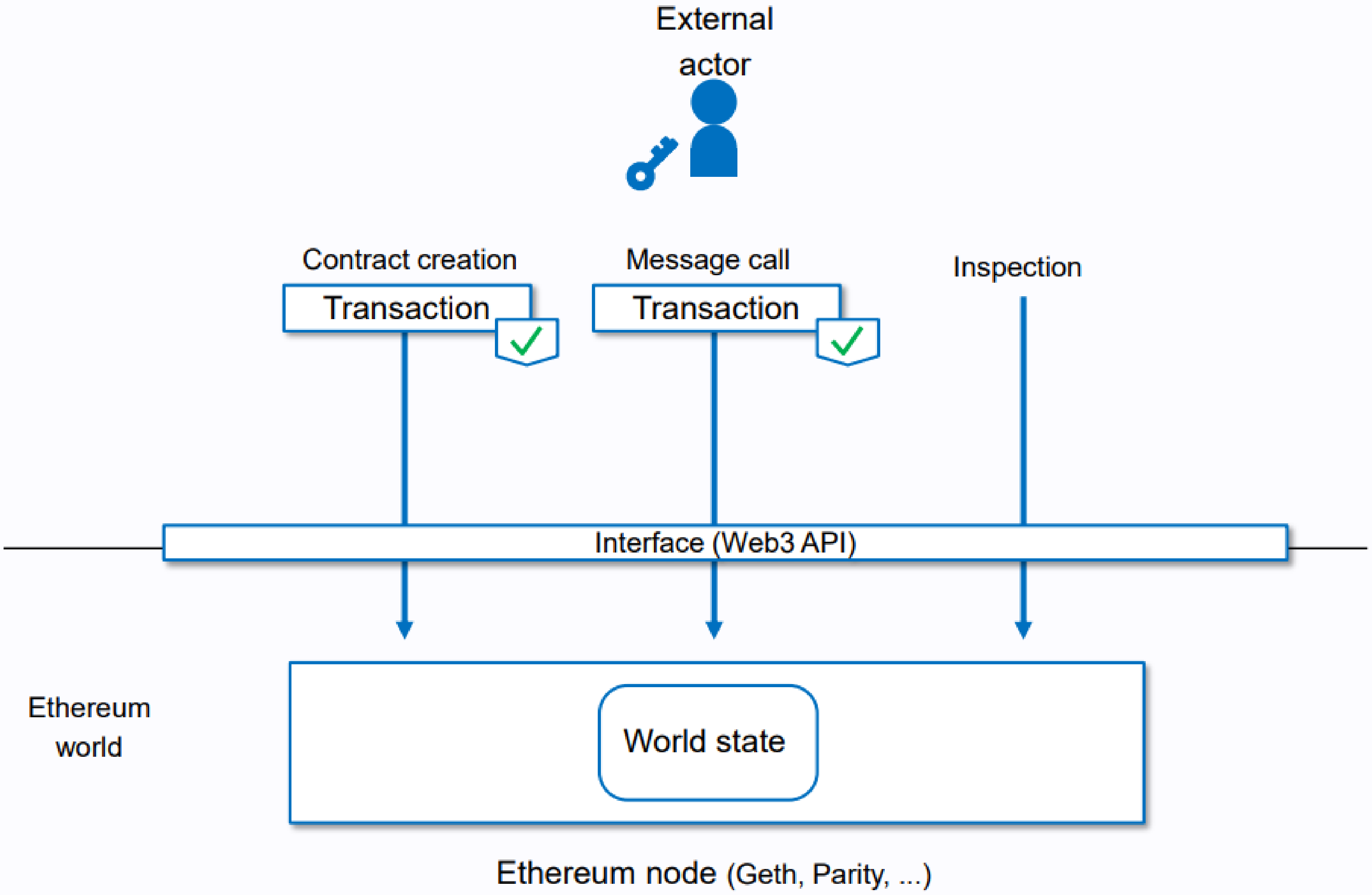


EIP-1559

Gas Cost = base fee + tip

Refund = max fee - (base fee + tip)

Interface to a node



External actor

Contract creation

Transaction

Message call

Transaction

Inspection

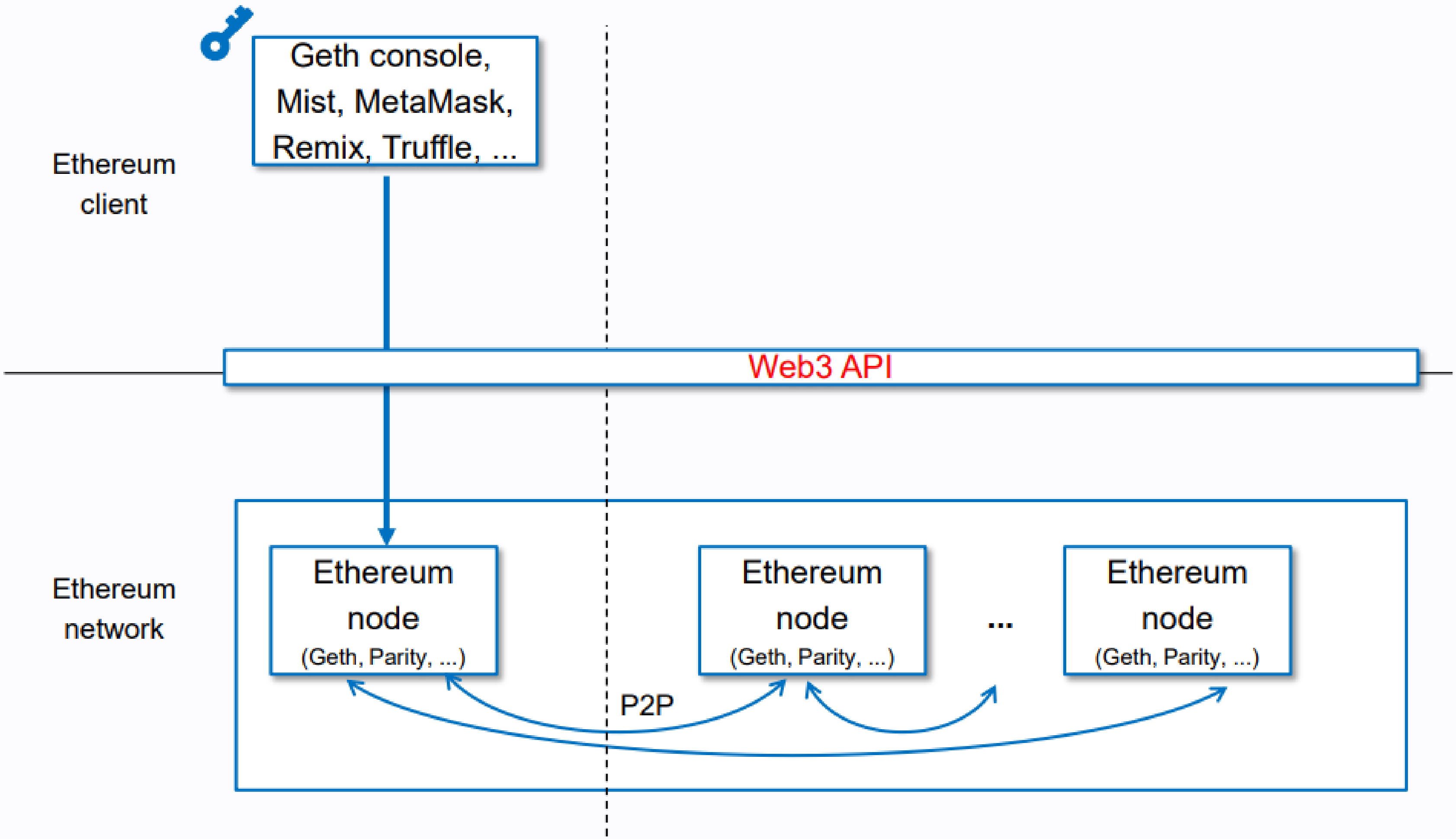
Interface (Web3 API)

Ethereum world

World state

Ethereum node (Geth, Parity, ...)

External actors access the Ethereum world through Ethereum nodes.



Geth console,
Mist, MetaMask,
Remix, Truffle, ...

Ethereum
client

Web3 API

Ethereum
network

Ethereum
node
(Geth, Parity, ...)

Ethereum
node
(Geth, Parity, ...)

...

Ethereum
node
(Geth, Parity, ...)

P2P

EIP

"Ethereum Improvement Proposals (EIPs) describe standards for the Ethereum platform, including core protocol specifications, client APIs, and contract standards."

-https://eips.ethereum.org/

EIP TYPES

EIPs are separated into a number of types, and each has its own list of EIPs:

- Standard Track (500)
- Core (189)
- Networking (13)
- Interface (42)
- ERC (256): Ethereum request for comment. These are application-level standards and conventions, including contract standards such as token standards (ERC20), and name registries (ERC137).
- Meta (18)
- Informational (6)

TOKENS

Many tokens are created using the ERC Smart Contract standards. Tokens can represent ownership of currencies or digital assets.

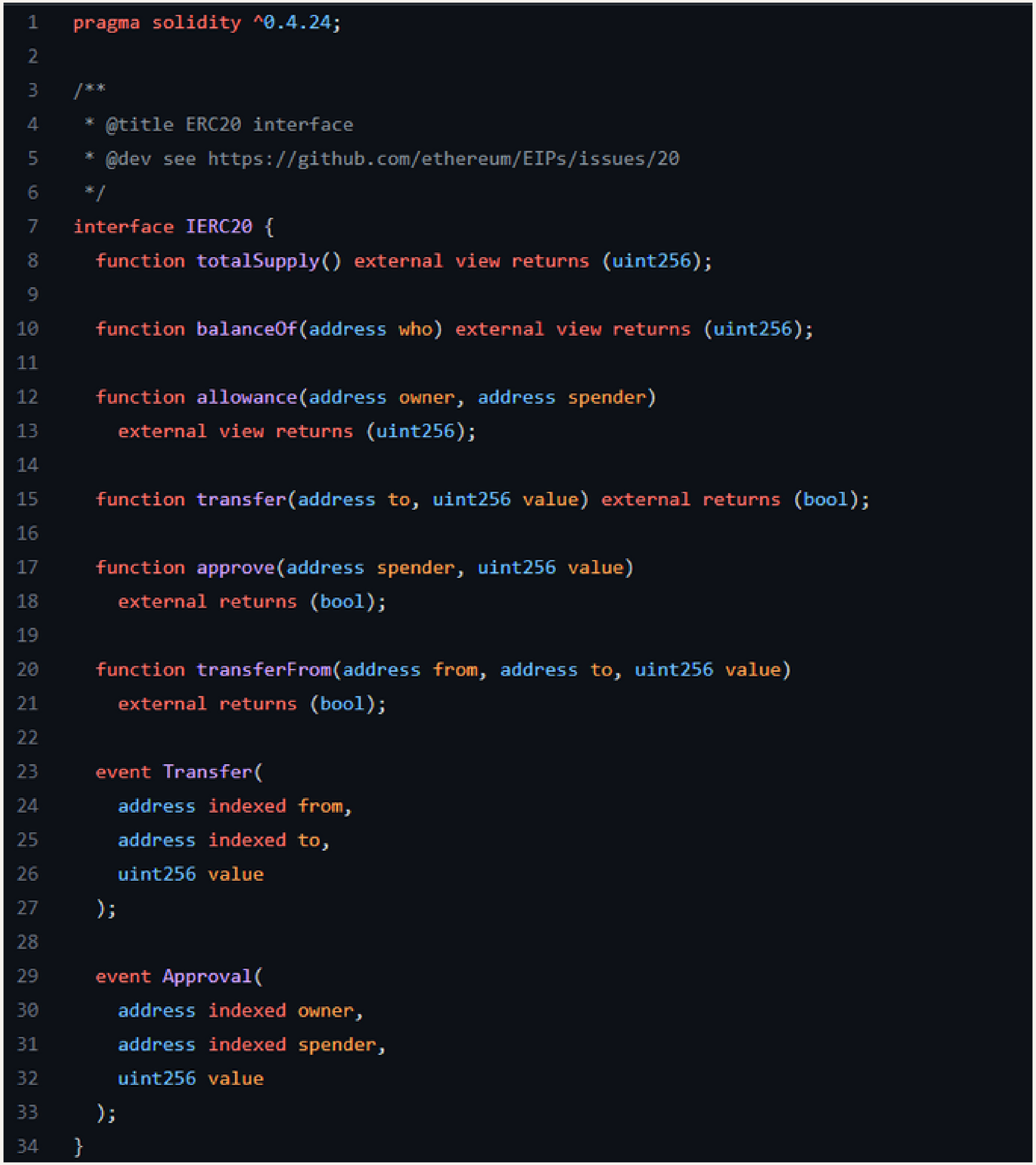
3 popular types of tokens:

- Fungible: Each commodity has the same value (Ex. Fiat Currency)
- Non-Fungible (NFT): Each commodity is unique (Ex. Driver's License)
- Semi-Fungible: Each set of commodity is unique (Ex. Pokemon Cards consisting of 5 Pikachus and 10 Charzards)

These tokens have widely accepted ERC standards:

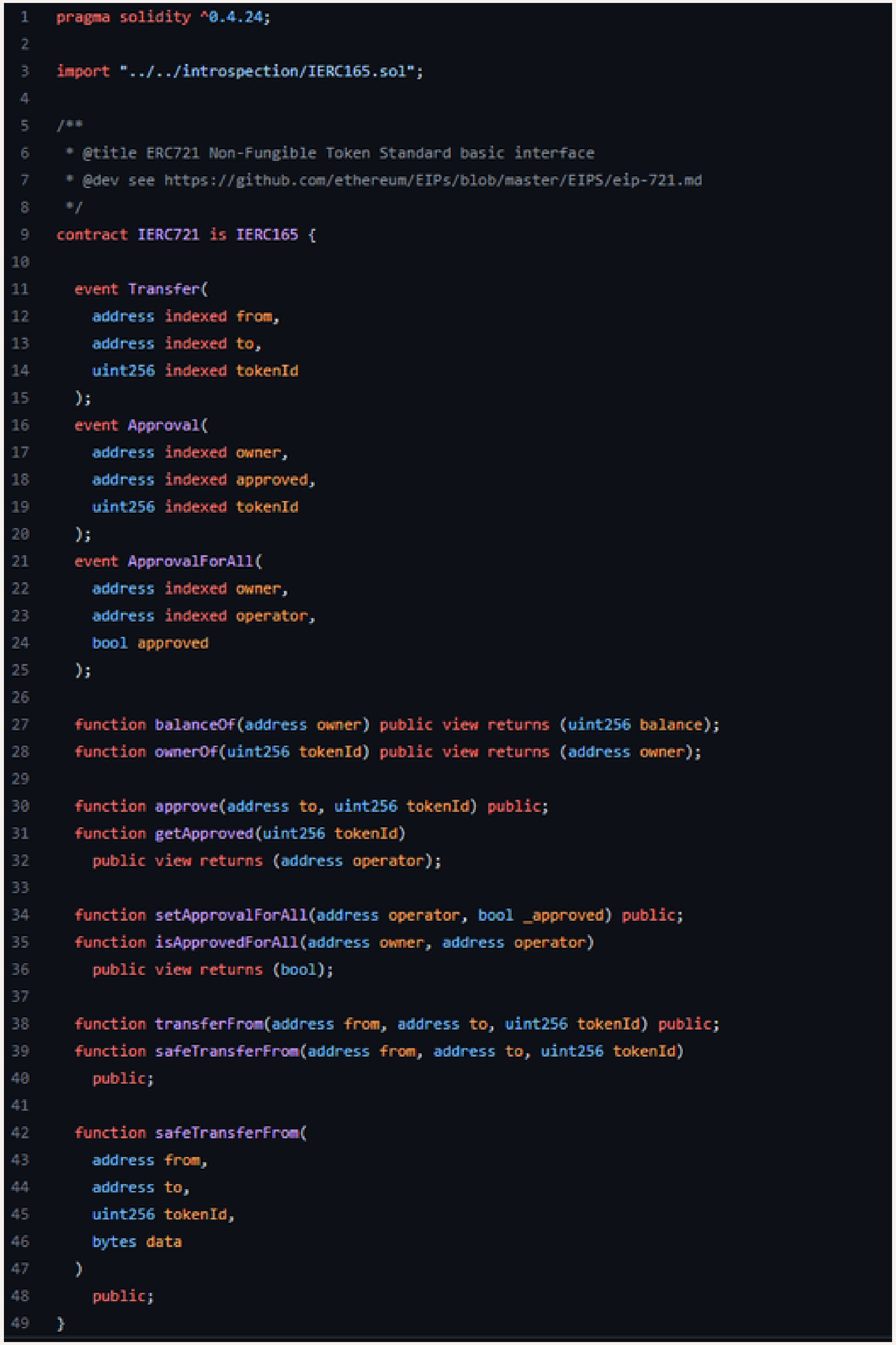
- ERC20: Fungible (Ex. Any token on Uniswap, except ETH)
- ERC721: NFT (Any token on Foundation NFT)
- ERC1155: Semi-Fungible, also known as "NFT" by the general community (Ex. Some tokens on Opensea)

ERC20



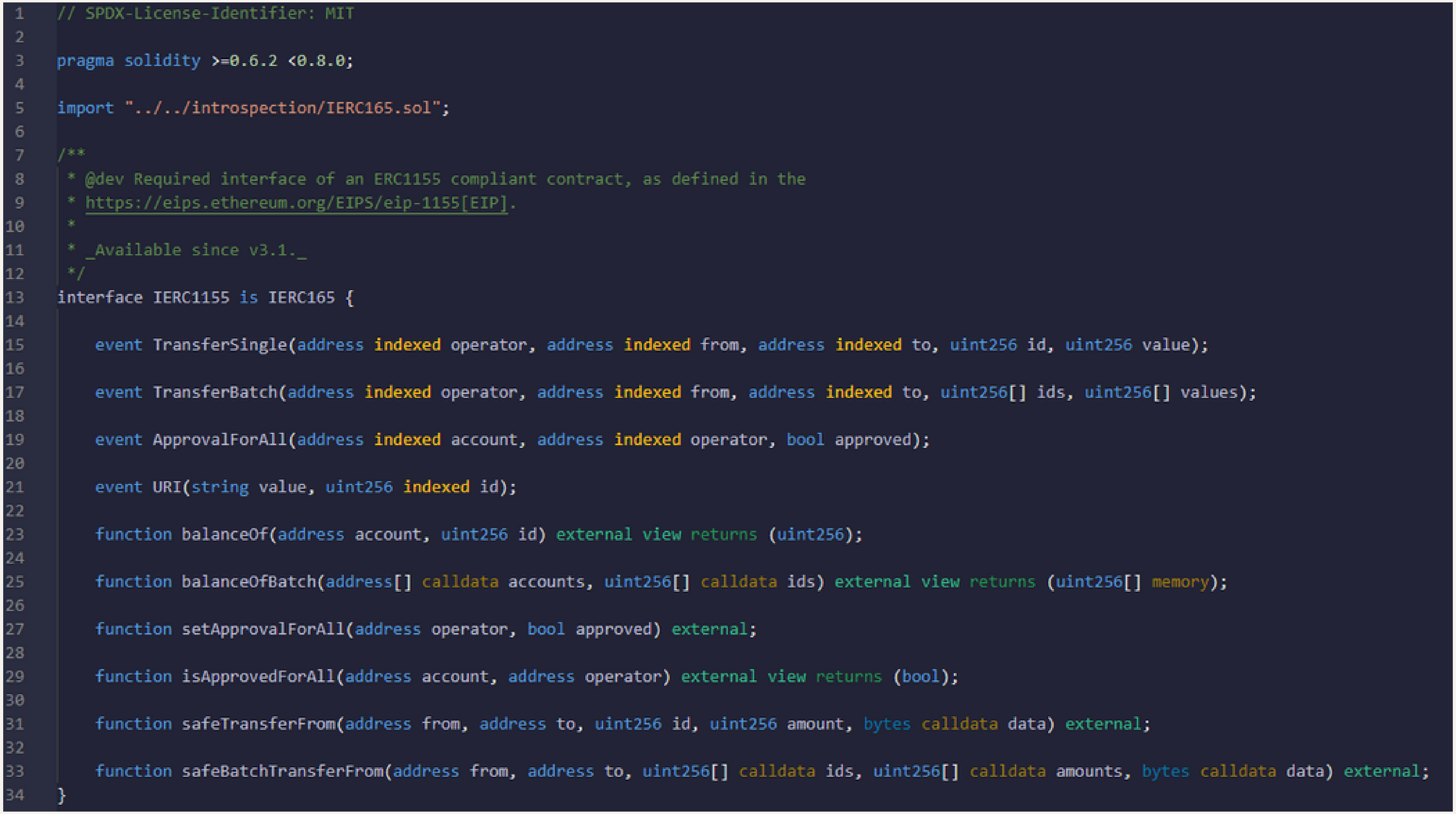
```solidity
pragma solidity ^0.4.24;

/**
 * @title ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/20
 */
interface IERC20 {
  function totalSupply() external view returns (uint256);

  function balanceOf(address who) external view returns (uint256);

  function allowance(address owner, address spender)
    external view returns (uint256);

  function transfer(address to, uint256 value) external returns (bool);

  function approve(address spender, uint256 value)
    external returns (bool);

  function transferFrom(address from, address to, uint256 value)
    external returns (bool);

  event Transfer(
    address indexed from,
    address indexed to,
    uint256 value
  );

  event Approval(
    address indexed owner,
    address indexed spender,
    uint256 value
  );
}
```

ERC721



```
pragma solidity ^0.4.24;

import "../../introspection/IERC165.sol";

/**
 * @title ERC721 Non-Fungible Token Standard basic interface
 * @dev see https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md
 */
contract IERC721 is IERC165 {

  event Transfer(
    address indexed from,
    address indexed to,
    uint256 indexed tokenId
  );
  event Approval(
    address indexed owner,
    address indexed approved,
    uint256 indexed tokenId
  );
  event ApprovalForAll(
    address indexed owner,
    address indexed operator,
    bool approved
  );

  function balanceOf(address owner) public view returns (uint256 balance);
  function ownerOf(uint256 tokenId) public view returns (address owner);

  function approve(address to, uint256 tokenId) public;
  function getApproved(uint256 tokenId)
    public view returns (address operator);

  function setApprovalForAll(address operator, bool _approved) public;
  function isApprovedForAll(address owner, address operator)
    public view returns (bool);

  function transferFrom(address from, address to, uint256 tokenId) public;
  function safeTransferFrom(address from, address to, uint256 tokenId)
    public;

  function safeTransferFrom(
    address from,
    address to,
    uint256 tokenId,
    bytes data
  )
    public;
}
```

ERC1155



```solidity
// SPDX-License-Identifier: MIT

pragma solidity >=0.6.2 <0.8.0;

import "../../introspection/IERC165.sol";

/**
 * @dev Required interface of an ERC1155 compliant contract, as defined in the
 * https://eips.ethereum.org/EIPS/eip-1155[EIP].
 *
 * _Available since v3.1._
 */
interface IERC1155 is IERC165 {

    event TransferSingle(address indexed operator, address indexed from, address indexed to, uint256 id, uint256 value);

    event TransferBatch(address indexed operator, address indexed from, address indexed to, uint256[] ids, uint256[] values);

    event ApprovalForAll(address indexed account, address indexed operator, bool approved);

    event URI(string value, uint256 indexed id);

    function balanceOf(address account, uint256 id) external view returns (uint256);

    function balanceOfBatch(address[] calldata accounts, uint256[] calldata ids) external view returns (uint256[] memory);

    function setApprovalForAll(address operator, bool approved) external;

    function isApprovedForAll(address account, address operator) external view returns (bool);

    function safeTransferFrom(address from, address to, uint256 id, uint256 amount, bytes calldata data) external;

    function safeBatchTransferFrom(address from, address to, uint256[] calldata ids, uint256[] calldata amounts, bytes calldata data) external;
}
```

TOKENIZED VAULTS

ERC-4626

An extension of ERC20 token standard that represents share ownership of an underlying asset.

KNOWING THE ECOSYSTEM

In blockchain, using deployed smart contract code and platform standards is encouraged because:

1. Avoid redundant development work
2. Security Audits and Testing already done
3. Lower learning curve for both Users and Developers due to similar Smart Contract APIs
4. Incorporate the larger amount of users and assets held by established dApps into a new dApp

KNOWING THE ECOSYSTEM

DeFi Example:

- ERC20
- Uniswap provides liquidity and facilitates trades for ERC20 tokens
- Web3 Startups make staking pools to incentivize people to provide liquidity to Uniswap for their token
- Staking Aggregators auto compound the staking rewards to earn a high APY
- Web3 Insurance offers automatic payout for Staking Aggregator smart contract hacks

Resources Used:

https://coinsbench.com/about-evm-opcode-gas-ethereum-accounts-9f0896f09d04
https://ethereum.org/
https://hardhat.org/
https://docs.ethers.io/v5/
https://www.openzeppelin.com/
https://takenobu-hs.github.io/downloads/ethereum_evm_illustrated.pdf
https://www.skillsoft.com/