BLOCKCHAIN

SMART CONTRACT AND TOKEN

Christine Hennebert

CEA LETI – Laboratoire des Systèmes Embarqués Sécurisés

ETHEREUM

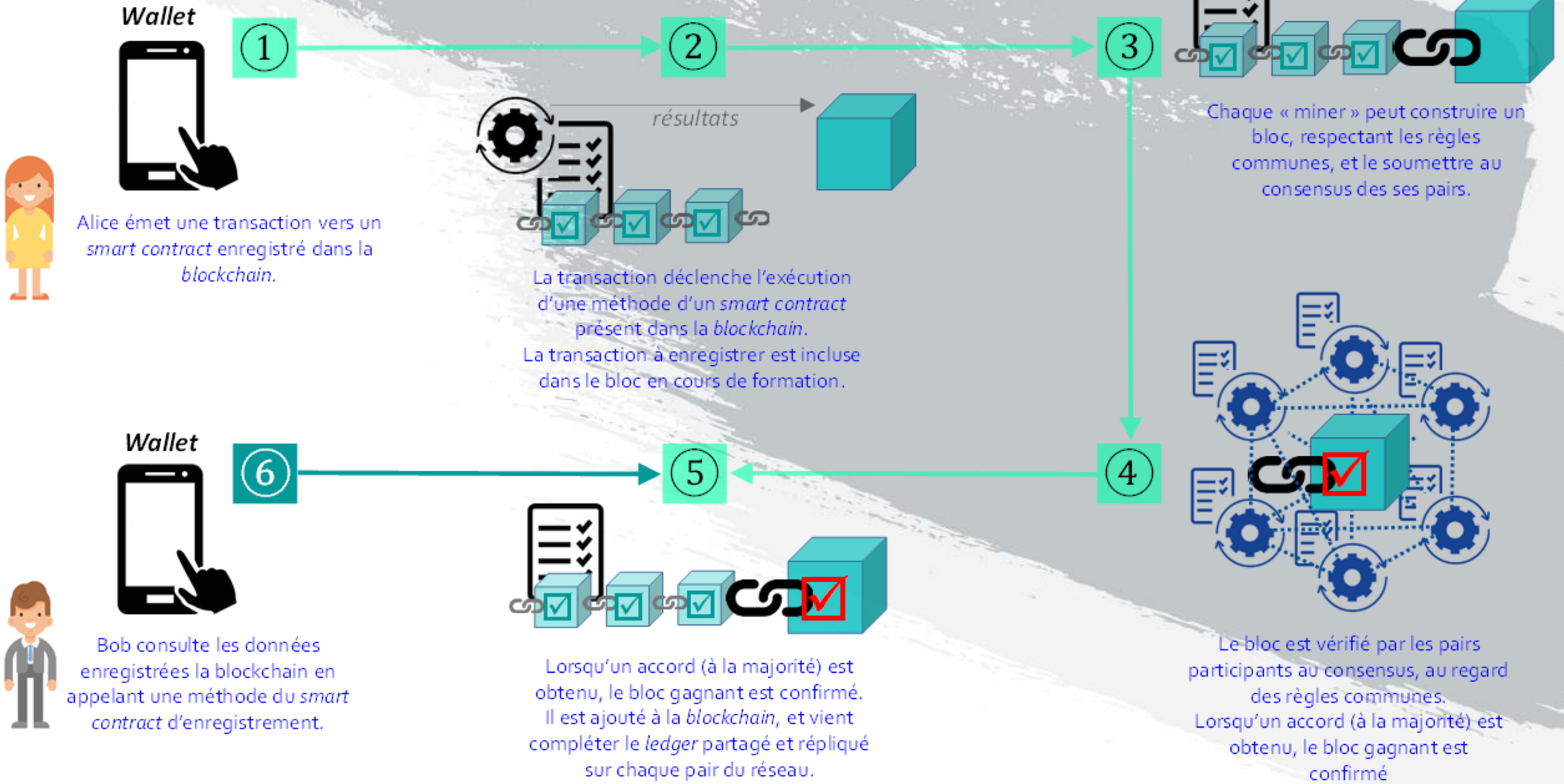
"Ethereum is an open blockchain platform that lets anyone build and use decentralized applications that run on blockchain technology".

Ethereum's official documentation

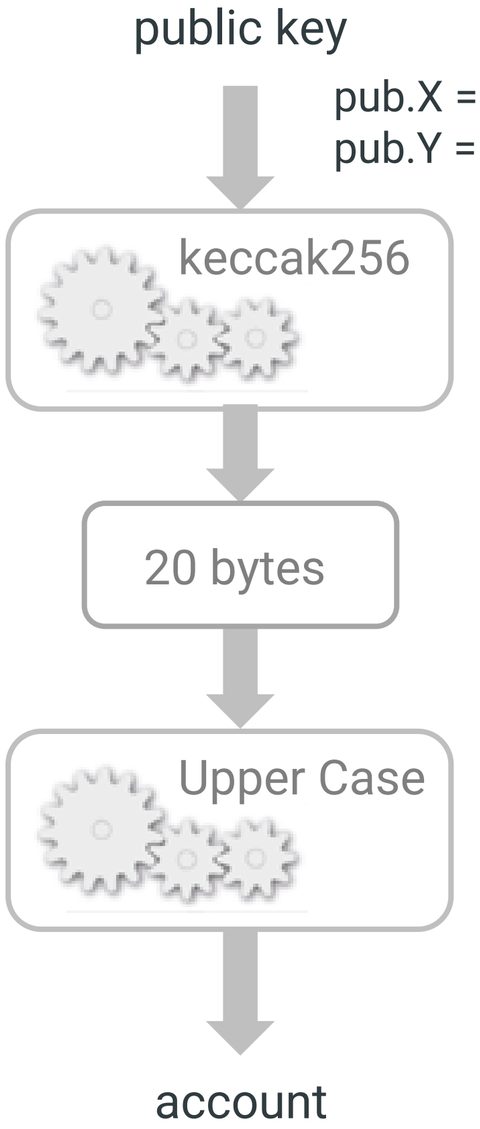
« the world computer »

Vitalik Buterin

HOW ETHEREUM WORKS?



ACCOUNT ADDRESS ETHEREUM



pub.X = 904D1127AD59B93C45F02A2F80E8FCA122746BC77AAA1416A6B0FE0C5D0982FD
pub.Y = 9AF40ADB96C0C92E59242E59EFBFDEF999FE120536C4B1F32F30BD8A1CF1ABEA

① Hash computing

② Selection of 20 bytes LSB

416CC7CC63FC93830750A3666A3C4BFD83B3F832

③ Computing and application of the mask (checksum)

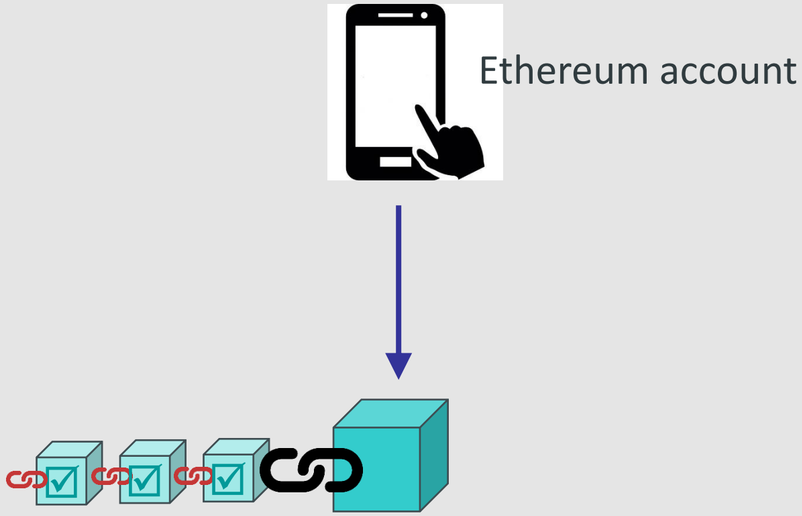
Keccak256(account address) → mask
For each quartet acc[i] in account : if mask[i] >=8 and account[i] is a letter → take upper case letter

④ Upper Case Account Address

account = 416CC7Cc63Fc93830750a3666a3C4bfD83b3F832

TWO TYPES OF TRANSACTIONS

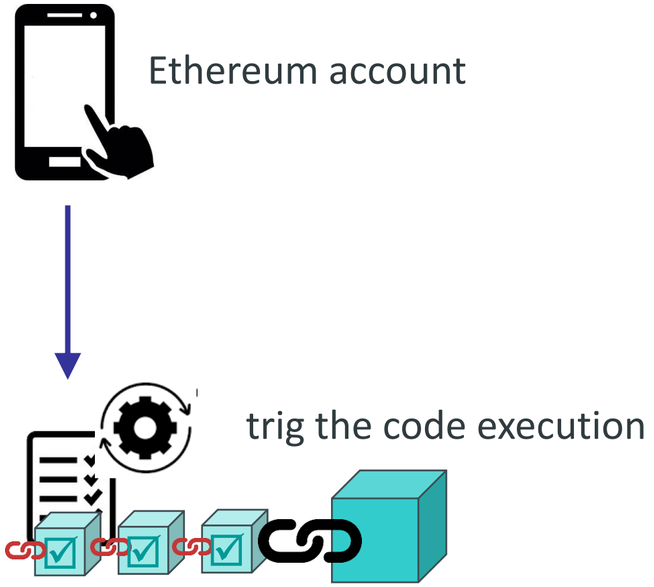
Transaction from account to account



change the account balance
source and destination account

```
tx = {
    'from': to_checksum_address(source),
    'to': to_checksum_address(destinataire),
    'value': value,
    'gasPrice': self.conn.gasPrice,
    'gas': 2 * self.conn.estimateGas(),
    'nonce': self.conn.eth_getTransactionCount(source),
    'chainId': 1
    }
```

Transaction from account to smart contract



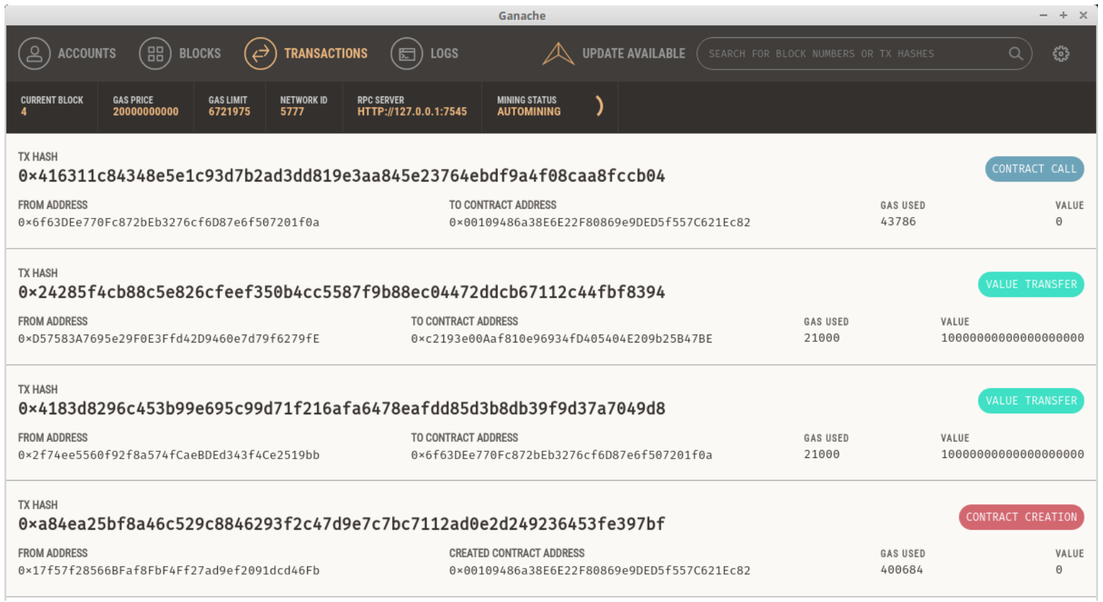

```
tx = {
    'value': 0,
    'from': to_checksum_address(source),
    'chainId': 1,
    'gas': self.conn.eth_estimateGas(source), ‘
    ‘to': to_checksum_address(contract_address),
    'gasPrice': self.conn.eth_gasPrice(),
    'nonce': self.conn.eth_getTransactionCount(source),
    'data': method | arguments
    }
```

EXEMPLE WITH GANACHE

Step ① : deployment of the smart contract

Step ② : Provisioning of the user account

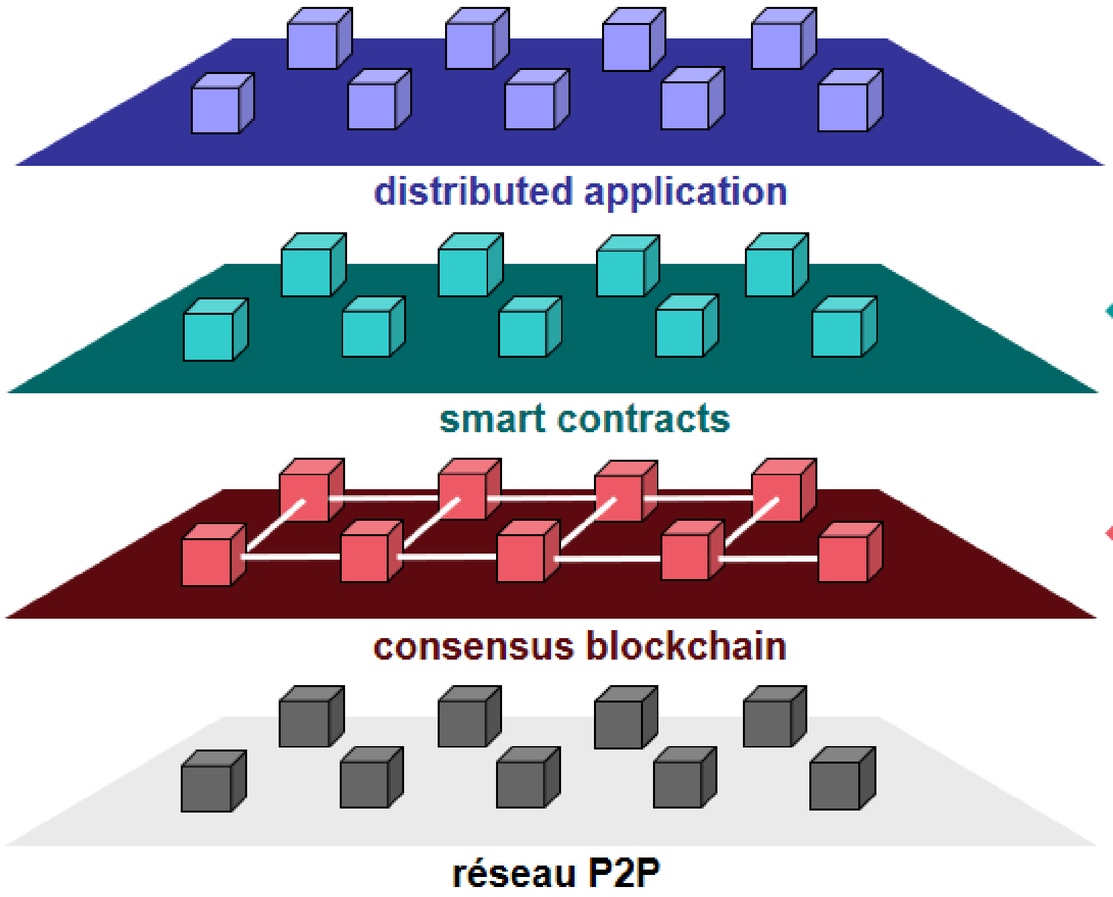
Step ③ : Send a transaction from the user account to the smart contract



CURRENCY VERSUS TOKEN

The code of the smart contract is executed by each validator node in a virtual machine (EVM)

The consensus is on the result of the execution



Token
A token is coded through a smart contract

Currency
The crypto-currency is created thanks to the incentive

CURRENCY = CRYPTO MONEY

Propriétés

Fongible : a unit is equivalent to another unit
Divisible : each unit can be divided into smaller units of value
Acceptable : widely accepted as mean of value exchange
Creation : an incentive mechanism enables to create unit
Limited quantity: the quantity in circulation is limited and/or capped
Portable : the units of value can be changed to another currency
Durable : the units can be used without being degraded

Usages

Money transfer
Store of value
Unit of account
… required to **deploy** **smart contracts**

TOKEN

digital representation of a value

Standard Token: **Ethereum Request for Comment (ERC)**

Fungible Token: standard ERC-20, optimized ERC-233, security token ERC-777…

- A fungible token is interchangeable with another token of the same type

Non Fungible Token: standard ERC-721

- A non-fungible token has unique characteristics and cannot be exchanged with another.
- Non-fungible tokens can have different values from one to another

Utility Token:

- may provide an access right
- enables access to an asset or a service provided via a blockchain

Security Token: Standard **ERC-1400**

- inherits the characteristics of fungible tokens **ERC-20** (or non fungible ERC-721)
- signifies the ownership of property or value
- is regulated in terms of identity, jurisdiction, or value used

TOKEN ERC20

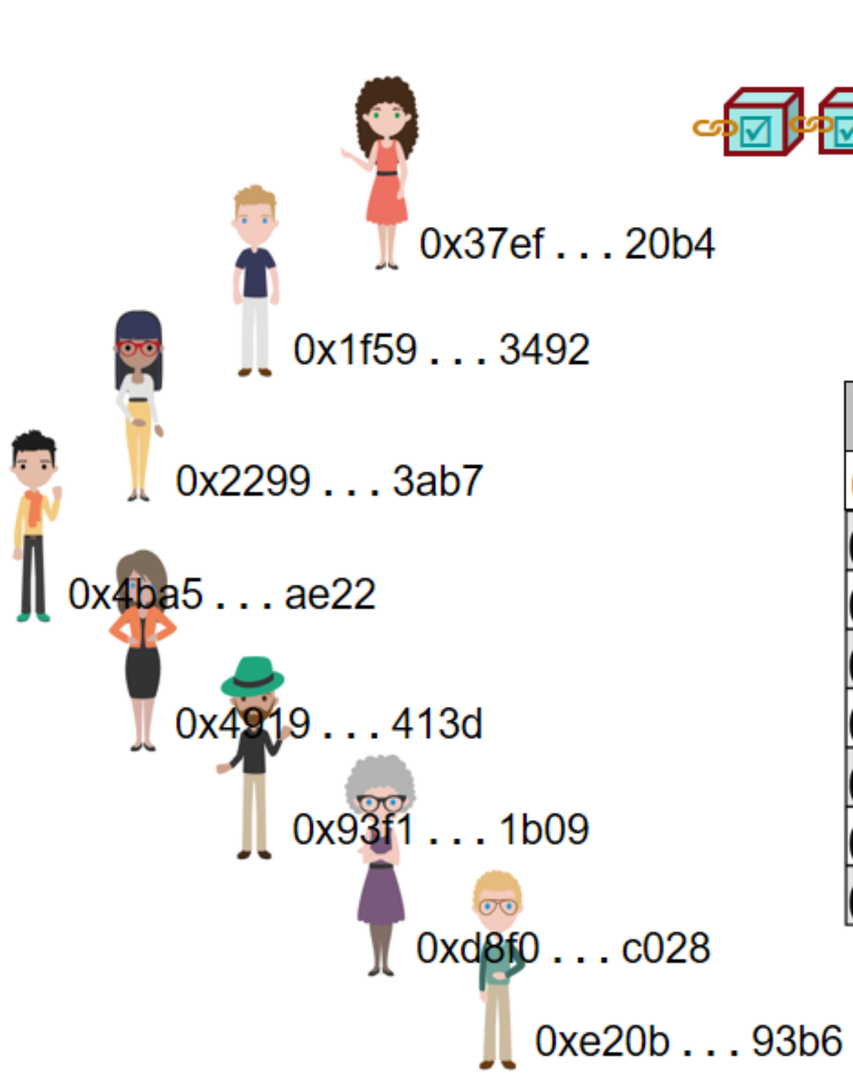
Ethereum Improvment Proposal (EIP) ≡ Implementation of ERC in **Solidity** language

https://eips.ethereum.org/EIPS/eip-20

- provides **the prototype of the standard functions** of the ERC20 token smart contract, which implements a record book accessible to users from their account address

- The reference implementations:
  - openzeppelin : https://github.com/OpenZeppelin/openzeppelin-contracts/
  - consensys : https://github.com/ConsenSys/Tokens/

TRANSFER FROM ACCOUNT TO ACCOUNT 1/2



smart contract ERC-20
« our-coin »
our-coin-address

holder account	balance
0x37ef . . . 20b4	100
0x1f59 . . . 3492	100
0x2299 . . . 3ab7	100
0x4ba5 . . . ae22	100
0x4919 . . . 413d	100
0x93f1 . . . 1b09	100
0xd8f0 . . . c028	100
0xe20b . . . 93b6	100
Total supply	**800**

TRANSFER FROM ACCOUNT TO ACCOUNT 2/2

0x1f59 . . . 3492

transfert(dst = 0x93f1…1b09, amount = 10)

balanceOf(account = 0xd8f0…c028)

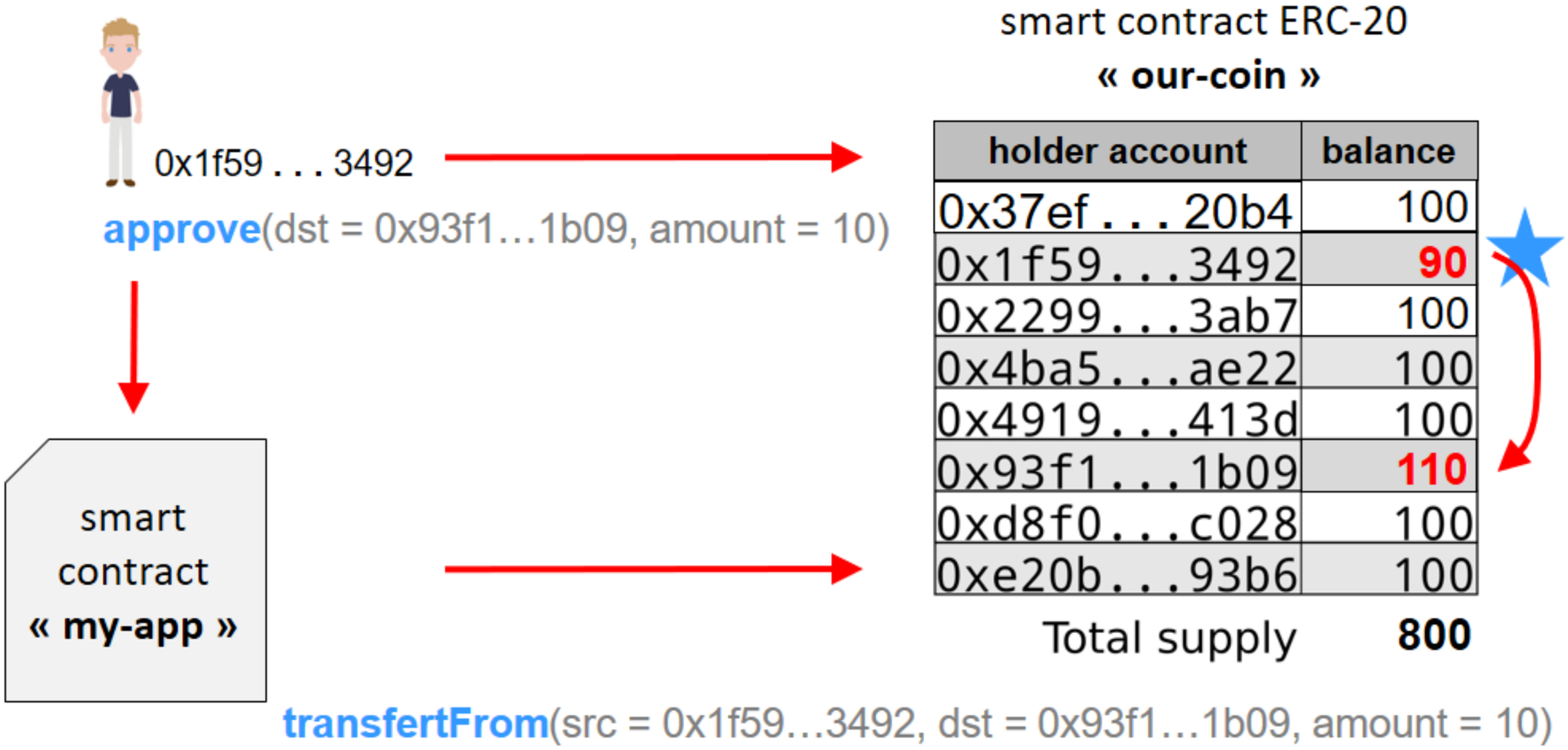
↵ **100**

smart contract ERC-20

« our-coin »

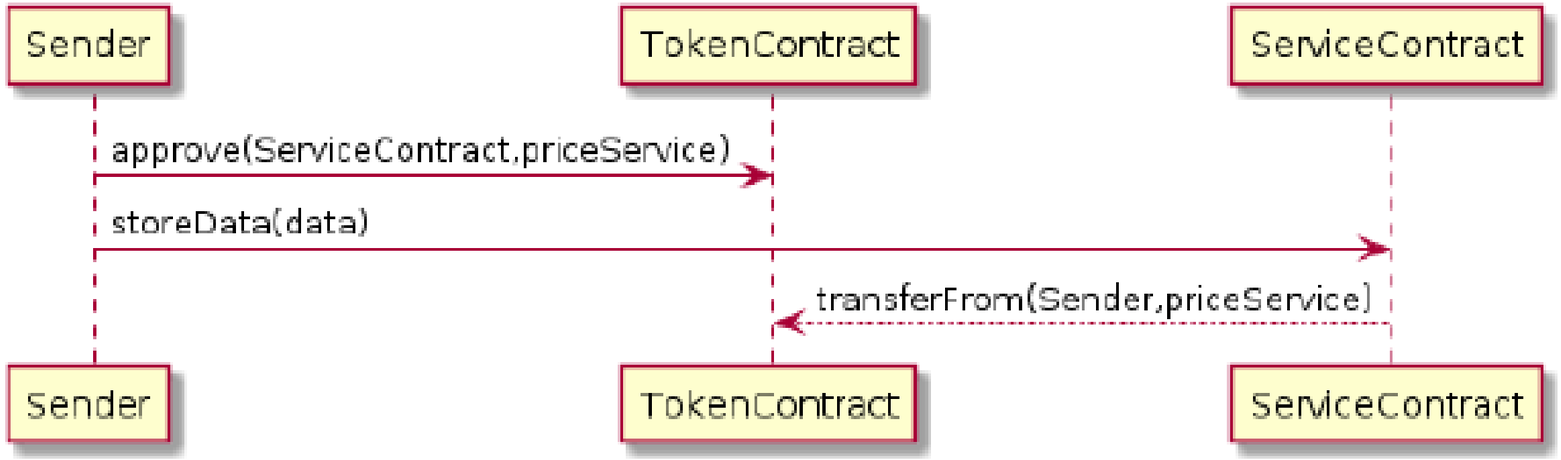
holder account	balance
0x37ef . . . 20b4	100
0x1f59 . . . 3492	**90**
0x2299 . . . 3ab7	100
0x4ba5 . . . ae22	100
0x4919 . . . 413d	100
0x93f1 . . . 1b09	**110**
0xd8f0 . . . c028	**100**
0xe20b . . . 93b6	100
Total supply	**800**

TRANSFER FROM ACCOUNT TO CONTRACT 1/2



holder account	balance
0x37ef . . . 20b4	100
0x1f59 . . . 3492	90
0x2299 . . . 3ab7	100
0x4ba5 . . . ae22	100
0x4919 . . . 413d	100
0x93f1 . . . 1b09	110
0xd8f0 . . . c028	100
0xe20b . . . 93b6	100

TRANSFER FROM ACCOUNT TO CONTRACT 2/2

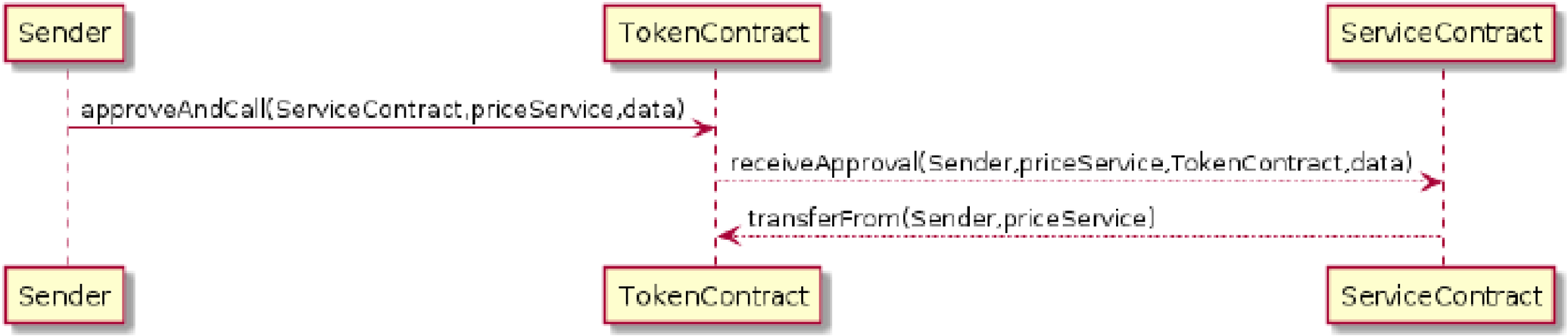


The user sends two transactions, which implies:

- To pay twice for gas,
- To introduce a vulnerability regarding either the possibility of being charged without the service being provided, or the service being provided without being charged,
- Hence the use of *requirement* in the service smart contract code.

```
function storeData( uint256 _data ) public{
                        require(tokenContract.transferFrom(msg.sender, receiver, 1));
                        dataBase[msg.sender] = _data;
                }
```

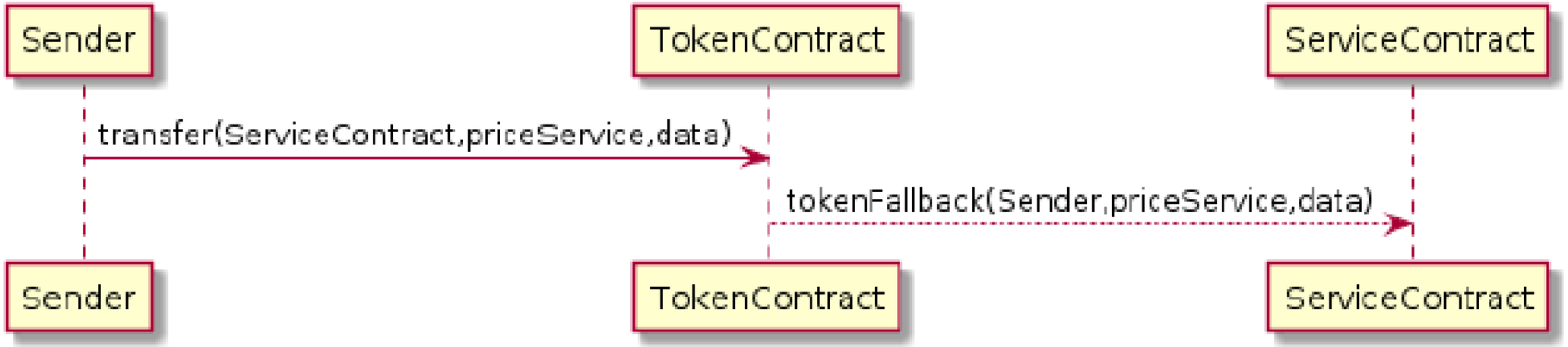
IMPROVED ERC20



An improvement to the standard ERC20 smart contract is to use the function **approveAndCall** :

- This function is not standard
- It is sent to the token smart contract, which calls the service and then debits the account
- The code of the function **approveAndCall** combines several functions, which makes it less optimal than using the two transactions in the previous case

TOKEN ERC223



With ERC223, the user sends only one transaction with the function **transfer** which is modified compared to the ERC20 reference implementation:

- **transfer** invokes the function **tokenFallBack** to initiate the inter-smart contract transaction to the service smart contract
- This enables less **gas** to be used than in the two previous cases.
- This potentially opens up a vulnerability depending on how carefully the code for the new function **transfer** is written : if the service fails, the account should not be debited, but the **gas** is lost

VULNERABILITIES

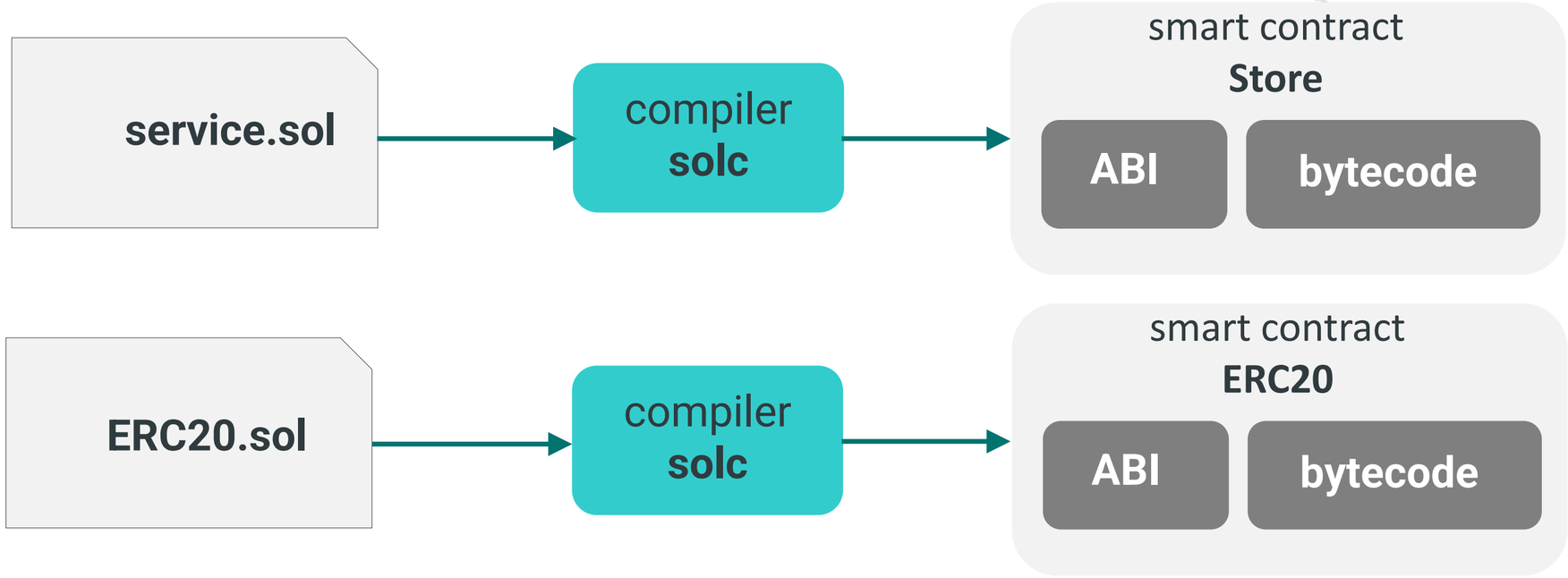
Smart Contract Weakness Classification and Test Cases

https://swcregistry.io/

Exemple

```
mapping (address => uint32) private _balances;

function _transfer(address sender, address recipient, uint32 amount) internal virtual {
        require(sender != address(0), "ERC20: transfer from the zero address");
        require(recipient != address(0), "ERC20: transfer to the zero address");

        _beforeTokenTransfer(sender, recipient, amount);

        _balances[sender] = _balances[sender] - amount;
        _balances[recipient] = _balances[recipient] + amount;
        emit Transfer(sender, recipient, amount);
    }
```

COMPILATION 1/2



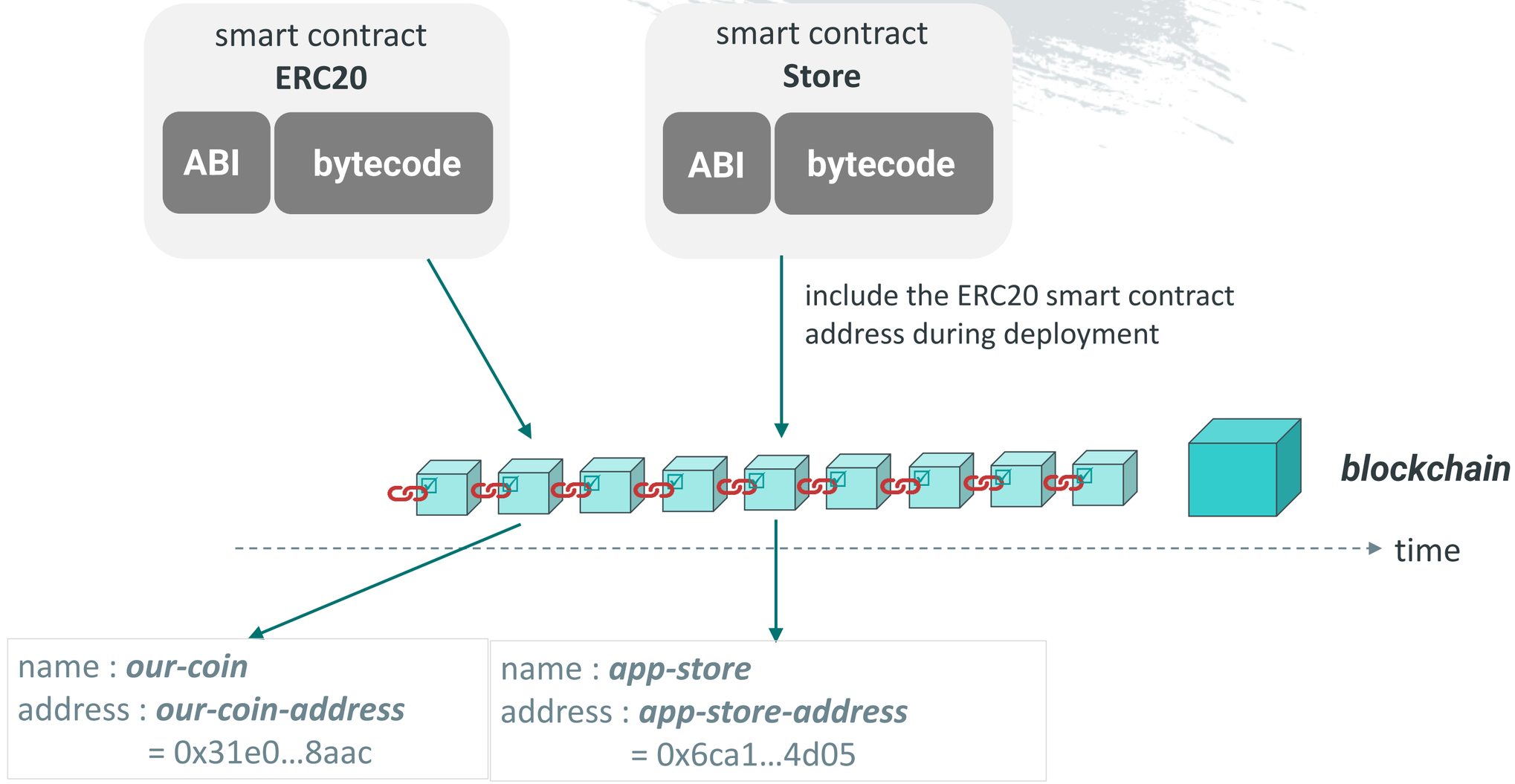
COMPILATION 2/2

```
BasicToken = pyeth.contracts.contract('BasicToken',
b'6080604052348015610010576000800fd5b506104178061002060003960000f3006080604052600436106100575760003357c0100000000
0000000000000000000000000000000000000000000000900463ffffffff16806318160ddd1461005c57806370a082311461008757806
a9059cbb146100de575b600080fd5b34801561006857600080fd5b506100716101435656b604051808281526020019150506040518091030
90f35b34801561009357600080fd5b506100c860048036038101908080357ffffffffffffffffffffffffffffffffffffffffffffffff1690602001909291905050505
061014d565b604051808281526020019150506040518091039f35b3480156100ea57600080fd5b506101296004803603810190808035773
fffffffffffffffffffffffffffffffffffffffff169060200190929190803590602001909291905050506101955656b604051808215151515815260200191
150506040518091039f35b60006001549050905656b60008060008373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff
ff16815260200190815260200160002054905091905056b60008073ffffffffffffffffffffffffffffffffffffffff168373ffffffffffffffffffffffffffffffffffffffff
fff16141515156101d257600080fd5b600803373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff1681526020019081
526020016000205482111515156102f1f57600080fd5b610270826000803373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff
ffffffffff16815260200190815260200160002054610 3b490919063ffffffff16565b6000803373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffff
ffffffffffffffffffffffff16815260200190815260200160002081905550610303826000808673ffffffffffffffffffffffffffffffffffffffff1673ffffffffffff
ffffffffffffffffffffffff16815260200190815260200160002054610 3cd90919063ffffffff16565b6000808573ffffffffffffffffffffffffffffffffffffffff
ff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002081905550827 3ffffffffffffffffffffffffffffffffffffffff163373ffffffffff
ffffffffffffffffffffffff167fddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef8460405180828152602001915
05060405180910390a3600190509291505056 5b60008282111515156103c257fe5b8183039050929150505656b6000808284019050838 1
101515156103e157fe5b8091505092915050560 0a165627a7a72305820cff2321f765033dd90f987a9d06d2bcd0cf0f6861d5c440616ec78
15c440aeb10029',
[{"constant":true,"inputs":[],"name":"totalSupply","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function
"},{"constant":true,"inputs":[{"name":"_owner","type":"address"}],"name":"balanceOf","outputs":[{"name":"balance","type":"uint256"}],"payable":fa
lse,"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"_to","type":"address"},{"name":"_value","type":"uint256"}],"nam
e":"transfer","outputs":[{"name":"","type":"bool"}],"payable":false,"stateMutability":"nonpayable","type":"function"},{"anonymous":false,"inputs":[{
"indexed":true,"name":"from","type":"address"},{"indexed":true,"name":"to","type":"address"},{"indexed":false,"name":"value","type":"uint256"}],"n
ame":"Transfer","type":"event"}])
```

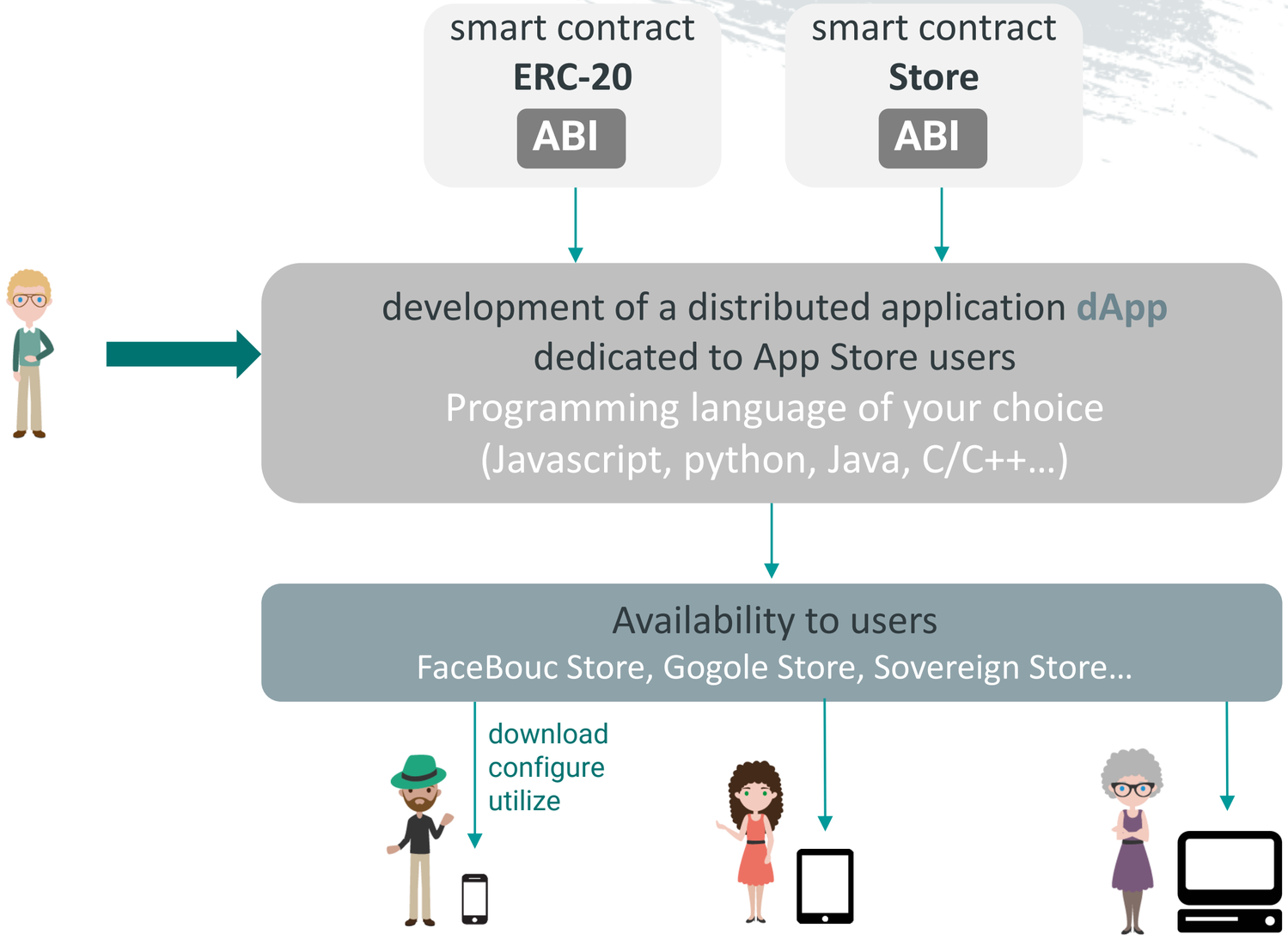
BasicToken.sol

```
pragma solidity ^0.4.18;

import "./ERC20Basic.sol";
import "../../math/SafeMath.sol";

contract BasicToken is ERC20Basic {

using SafeMath for uint256;

mapping(address => uint256) balances;
uint256 totalSupply_;

function totalSupply() public view returns (uint256) {
                return totalSupply_;
                }

function transfer(address _to, uint256 _value) public returns (bool) {
                require(_to != address(0));    require(_value <= balances[msg.sender]);
                balances[msg.sender] = balances[msg.sender].sub(_value);
                balances[_to] = balances[_to].add(_value);
                Transfer(msg.sender, _to, _value);
                return true;
                }

function balanceOf(address _owner) public view returns (uint256 balance) {
                return balances[_owner];
                }
}
```

DEPLOYMENT 1/2



DEPLOYMENT 2/2



SOME REFERENCES

1. library openzeppelin, https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/token
2. library consensys, https://github.com/ConsenSys/Tokens/
3. Solidity readthedocs, https://docs.soliditylang.org/en/latest/
4. Solidity by example, https://docs.soliditylang.org/en/latest/solidity-by-example.html
5. Solidity github, https://github.com/ethereum/solidity/
6. Vyper readthedocs, https://vyper.readthedocs.io/en/stable/
7. Vyper by example, https://vyper.readthedocs.io/en/latest/vyper-by-example.html
8. Vyper github, https://github.com/vyperlang/vyper