
# BLOCKCHAIN

## SMART CONTRACT ET TOKEN

Christine Hennebert

CEA LETI – Laboratoire des Systèmes Embarqués Sécurisés

# ETHEREUM

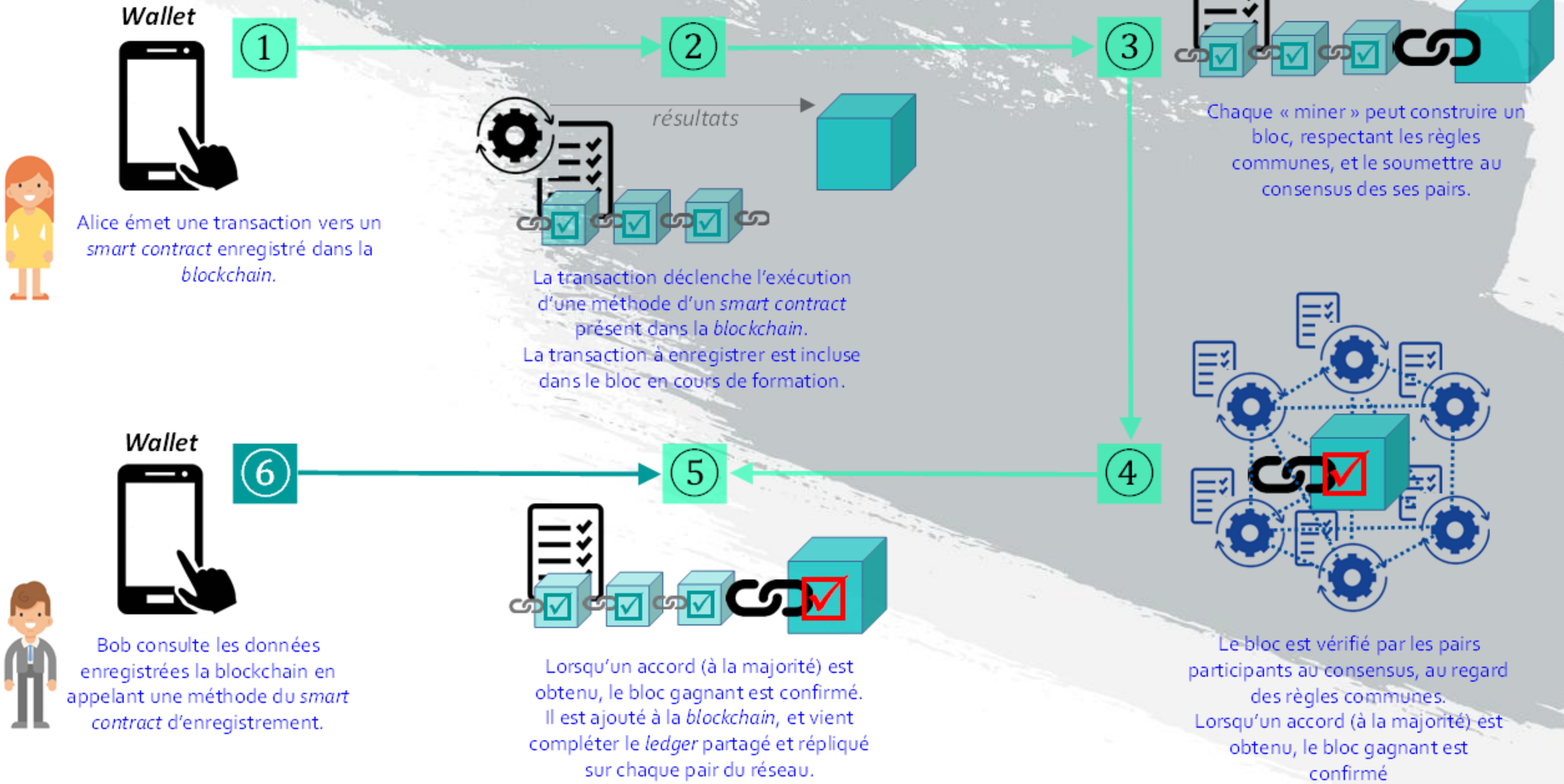
***"Ethereum is an open blockchain platform that lets anyone build and use decentralized applications that run on blockchain technology".***

Ethereum's official documentation

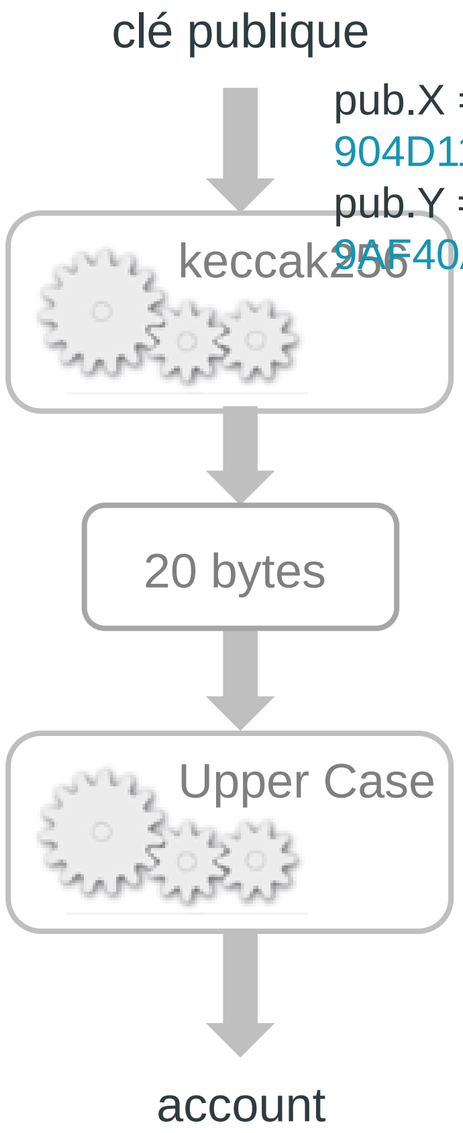
***« the world computer »***

Vitalik Buterin

# COMMENT FONCTIONNE ETHEREUM ?



# ADRESSE DE COMPTE ETHEREUM




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

① Calcul du haché

② Sélection des 20 octets LSB

416CC7CC63FC93830750A3666A3C4BFD83B3F832

③ Calcul et Application du masque (checksum)

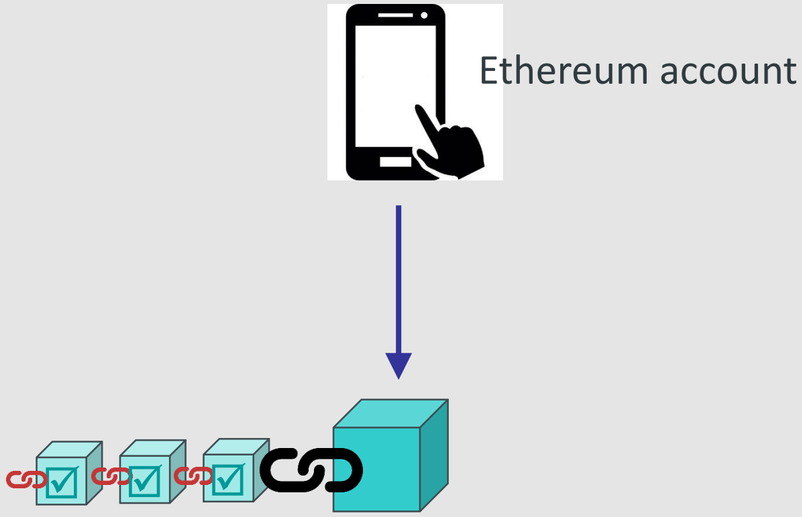
Keccak256(account address) → mask
For each quartet acc[i] in account : if mask[i] >=8 and account[i] is a letter → take upper case letter

④ Upper Case Account Address

account = 416CC7Cc63Fc93830750a3666a3C4bfD83b3F832

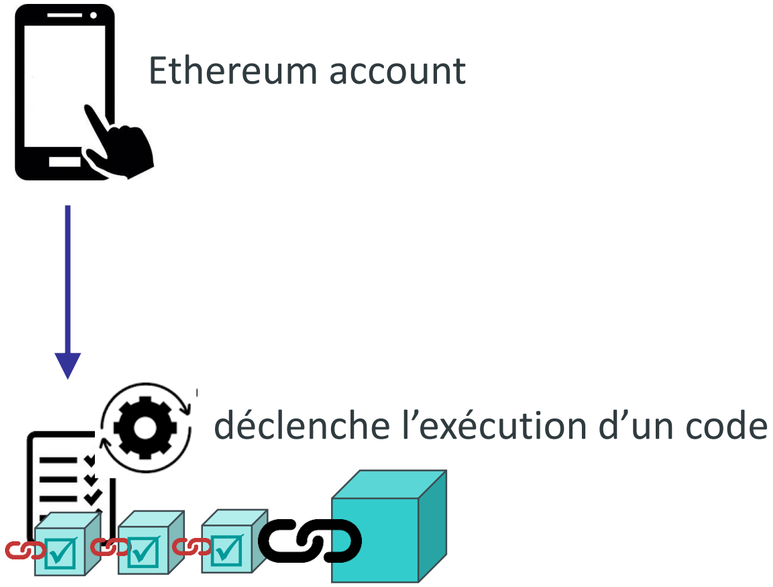
# DEUX TYPES DE TRANSACTIONS

## Transaction de compte à compte




```
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 de compte à smart contract



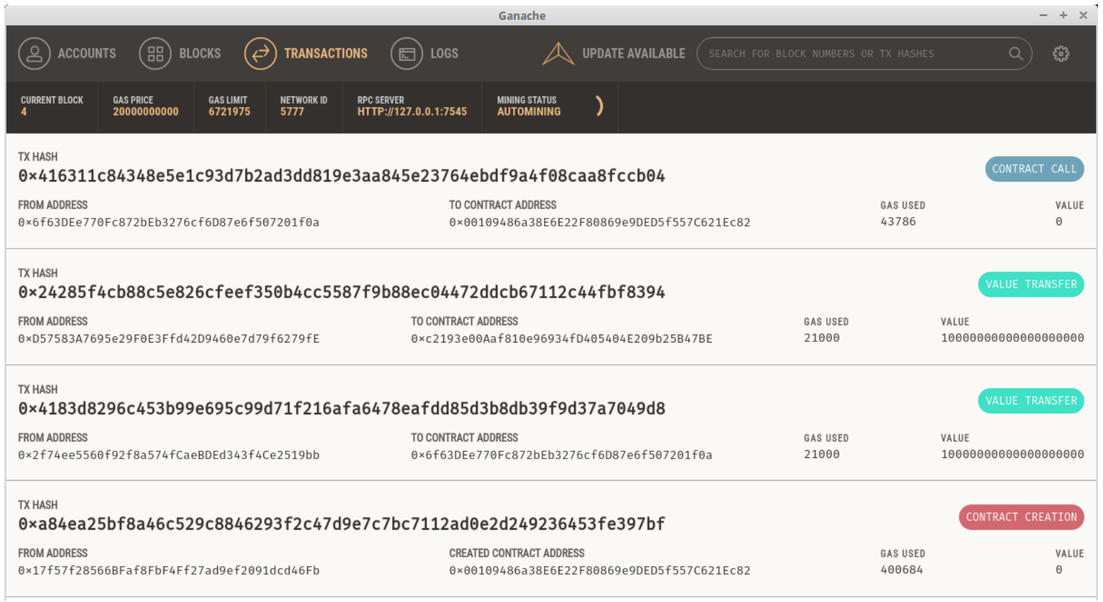

```
tx = {
    'value': 0,
    'from': to_checksum_address(source),
    'chainId': 1,
    'gas': self.conn.eth_estimateGas(source), ‘
    ‘to': to_checksum_address(contract_address),
    'gasPrice': self.conn.eth_gasPrice(),
    'nonce': self.conn.eth_getTransactionCount(source),
    'data': method |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

# EXEMPLE AVEC GANACHE

**Etape ①** : déploiement du smart contract

**Etape ②** : Approvisionnement du compte utilisateur

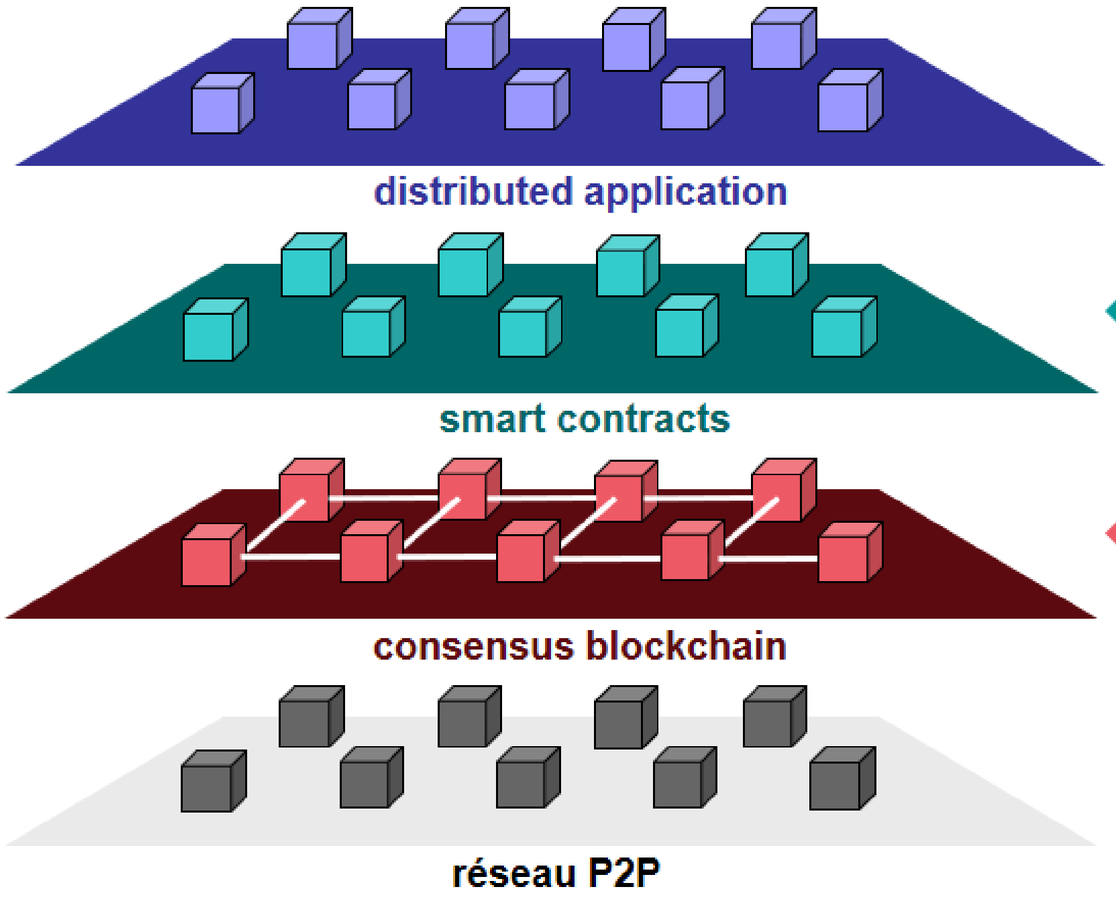
**Etape ③** : Envoi d'une transaction depuis le compte utilisateur vers le smart contract



# CURRENCY VERSUS TOKEN

Le code du smart contract est exécuté par chaque nœud validateur dans une machine virtuelle (EVM)

Le consensus porte sur la résultat de l'exécution




**Token**
Programmation d'un token avec un Smart Contract

**Currency**
Création de crypto-monnaie grâce à l'incitation

# CURRENCY = CRYPTO MONNAIE

## Propriétés

**Fongible** : une unité est équivalente à une autre
**Divisible** : chaque unité peut être divisée en petites unités de valeur
**Acceptable** : largement accepté comme moyen d'échange
**Création** : un mécanisme d’incentive permet de créer les unités
**Quantité limitée** : la quantité d’unités en circulation est limitée et/ou plafonnée
**Portable** : les unités peuvent être échangées contre une autre currency
**Durable** : les unités peuvent être utilisées sans être dégradées

## Usages

**Transfert d’argent**
**Réserve de valeurs**
**Unité de compte**
… pour **déployer** des **smart contracts**

# TOKEN

## représentation numérique d'une valeur

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

**Token fongible** : standard ERC-20, optimisé ERC-233, sécuritaire ERC-777…

- un token fongible est interchangeable avec un autre token du même type

**Token non fongible** : standard ERC-721

- un token non fongible présente des caractéristiques uniques, et ne peut pas être échangé avec un autre.
- les tokens non fongibles peuvent avoir des valeurs différentes les uns des autres.

**Utility Token** :

- peut représenter un droit d'accès
- permet d'accéder à un bien ou à un service fourni via une blockchain

**Security Token** : Standard **ERC-1400**

- hérite des caractéristiques des token fongible **ERC-20** (ou non fongible ERC-721)
- représente la propriété d'un bien ou d'une valeur
- est régulé quant à l'identité, la juridiction ou la valeur utilisée

# TOKEN ERC20

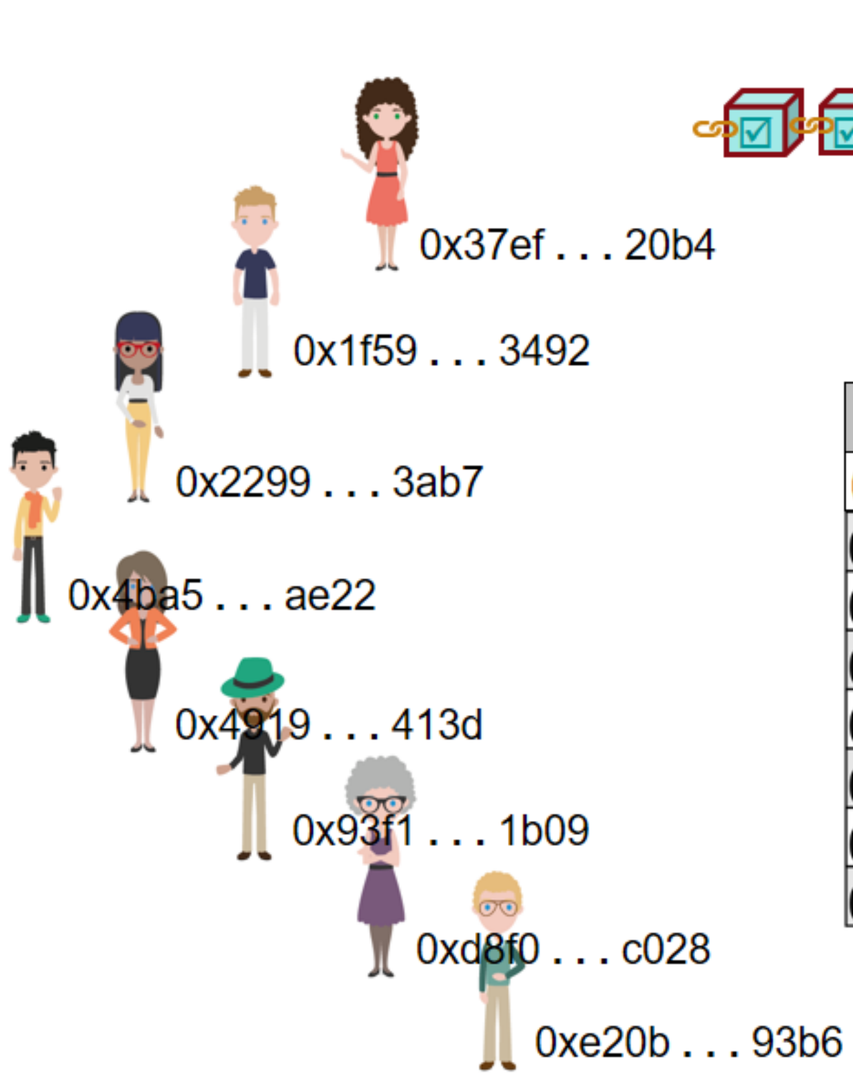
**Ethereum Improvment Proposal (EIP)** ≡ Implémentation des ERC en langage **Solidity**

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

- fournit le **prototype des fonctions standards** du smart contract de token ERC20 qui implémente un livre de compte accessible aux utilisateurs depuis leur adresse de compte

- les **implémentations de référence** :
  - openzeppelin : https://github.com/OpenZeppelin/openzeppelin-contracts/
  - consensys : https://github.com/ConsenSys/Tokens/

# TRANSFERT DE COMPTE À COMPTE 1/2




***blockchain***

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** |

# TRANSFERT DE COMPTE À COMPTE 2/2

0x1f59 . . . 3492

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

**balanceOf**(account = 0xd8f0…c028)

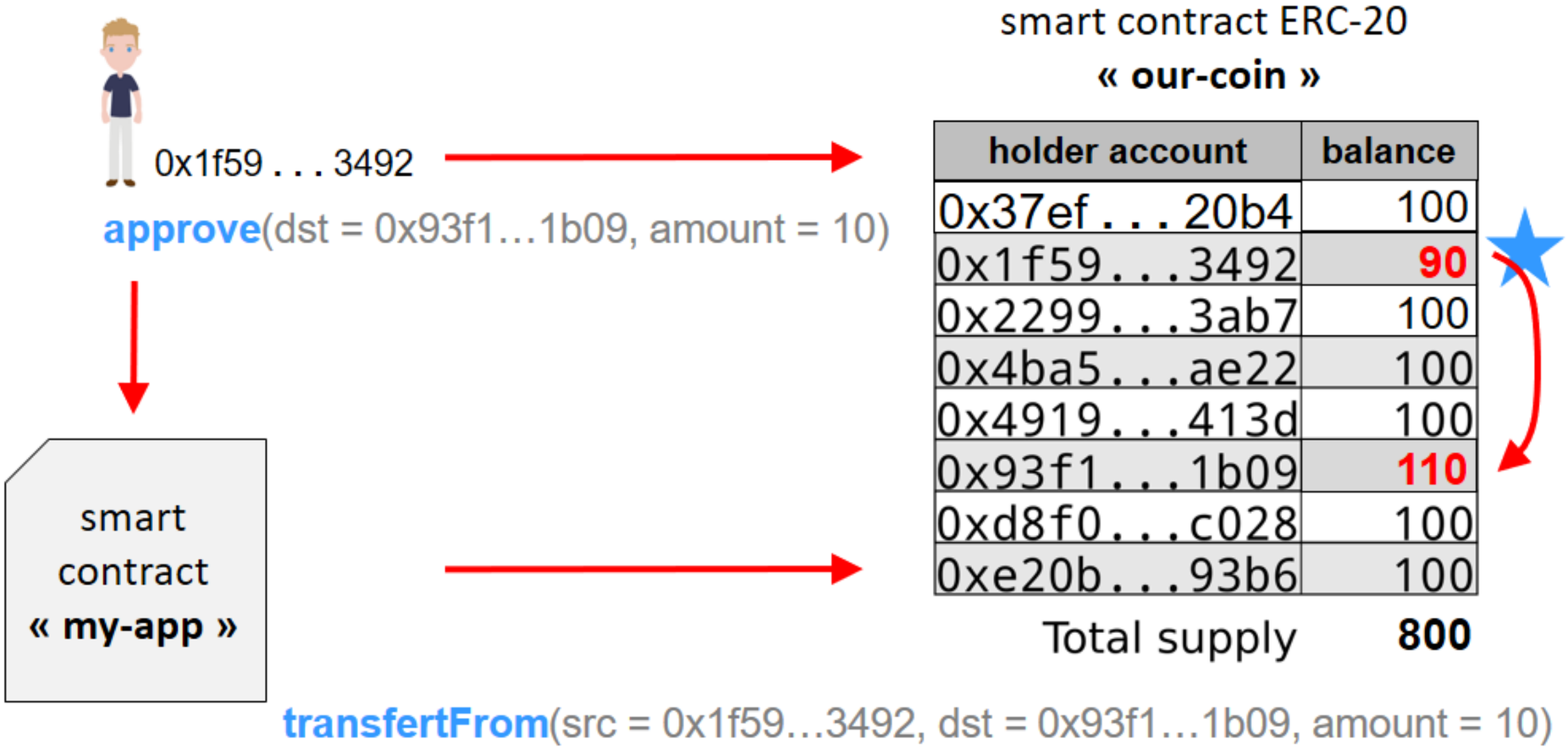
↵ **100**

smart contract ERC-20

**« our-coin »**

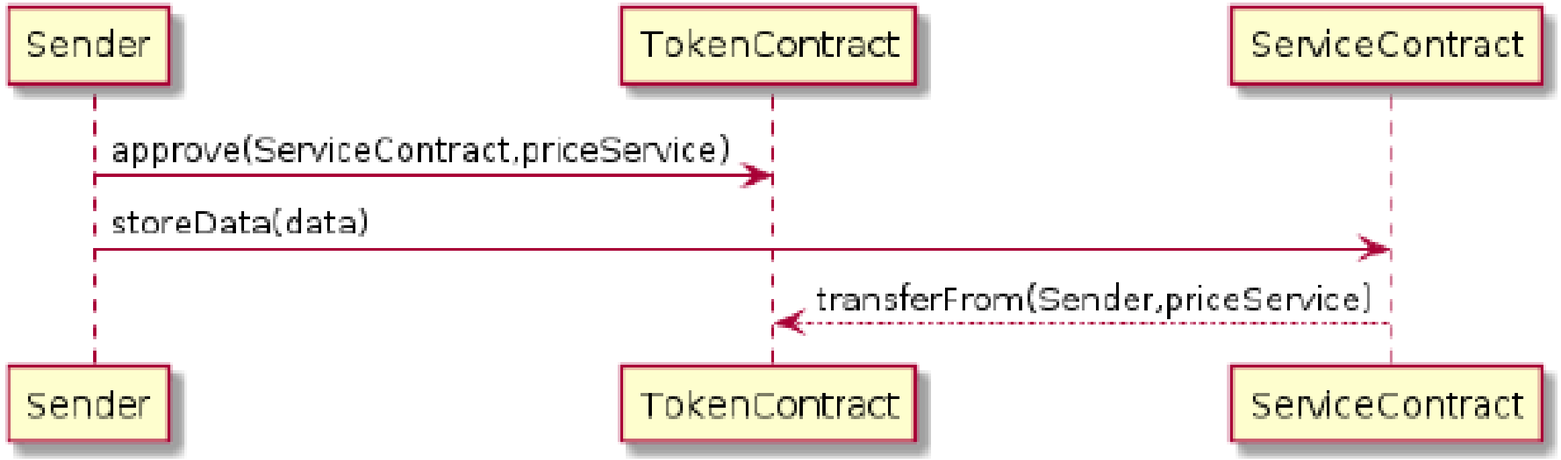
| holder account | balance |
|---|---|
| 0x37ef . . . 20b4 | 100 |
| 0x1f59 . . . 3492 | **90** |
| 0x2299 . . . 3ab7 | 100 |
| 0x4ba5 . . . ae22 | 100 |
| 0x4919 . . . 413d | 100 |
| 0x93f1 . . . 1b09 | **110** |
| 0xd8f0 . . . c028 | **100** |
| 0xe20b . . . 93b6 | 100 |
| Total supply | **800** |

# TRANSFERT DE COMPTE À 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 |

# TRANSFERT DE COMPTE À CONTRACT 2/2

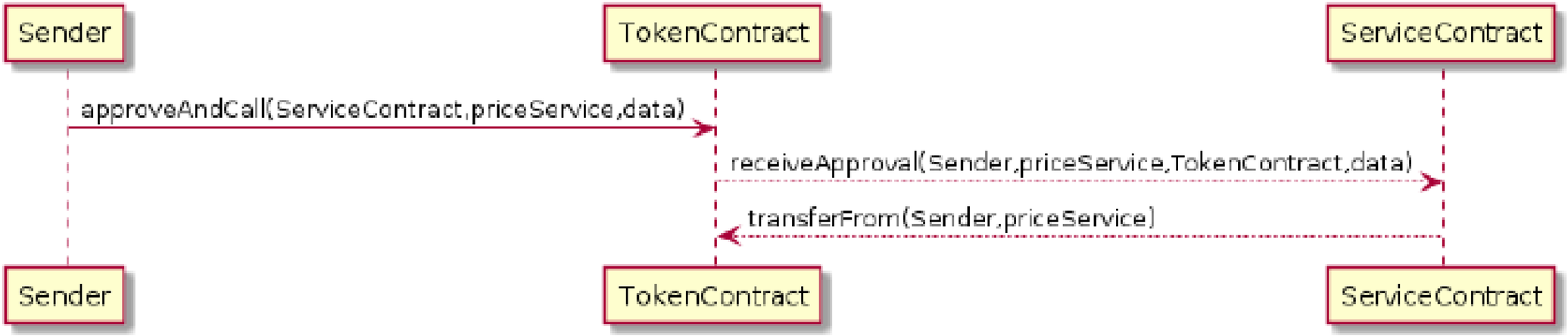



L'utilisateur envoie deux transactions, ce qui implique :

- De payer deux fois du gas
- D'ouvrir une faille quant à la possibilité d'être débiter sans que le service soit effectué, ou que le service soit effectué sans être débité
- D'où l'usage de *requirement* dans le code du smart contract de service

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

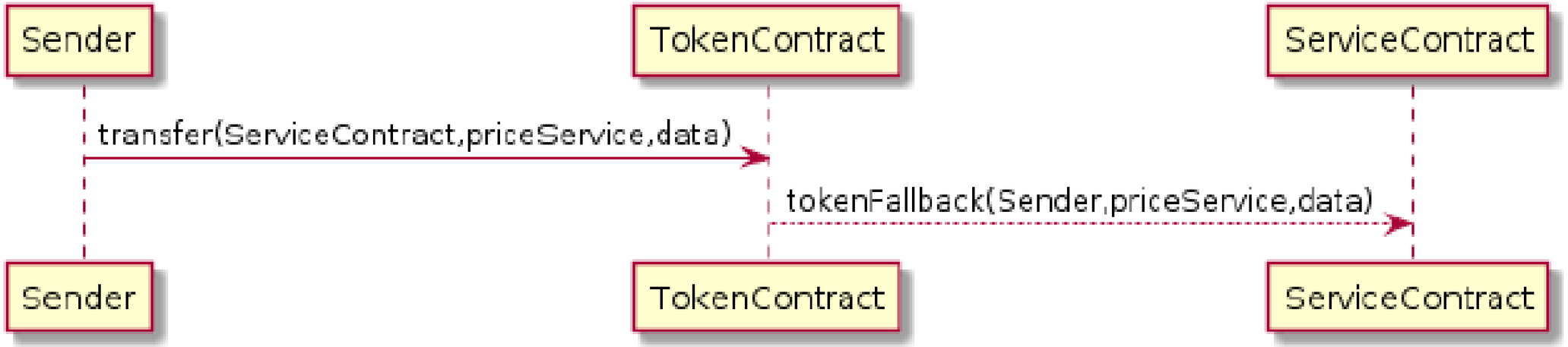
# ERC20 AMÉLIORÉ




Une amélioration au smart contract standard ERC20 consiste à utiliser la fonction **approveAndCall** :

- Cette fonction n'est pas standard
- Elle est envoyée vers le smart contract de token qui appelle le service puis débite le compte
- Le code de la fonction **approveAndCall** imbrique plusieurs fonctions, ce qui le rend moins optimal que de faire appel aux deux transactions du cas précédent

# TOKEN ERC223




Avec ERC223 l'utilisateur n'envoie qu'une seule transaction avec la fonction **transfer** qui est modifiée par rapport à l'implémentation de référence ERC20 :

- **transfer** fait appel à la fonction **tokenFallBack** pour lancer la transaction inter smart contract vers le smart contract de service
- Cela permet de dépenser moins de **gas** que dans les deux cas précédents
- Cela ouvre potentiellement une faille selon le soin apporté au code de la nouvelle fonction **transfer** : si le service échoue, le compte ne doit pas être débité, mais le **gas** est perdu

# VULNÉRABILITÉS

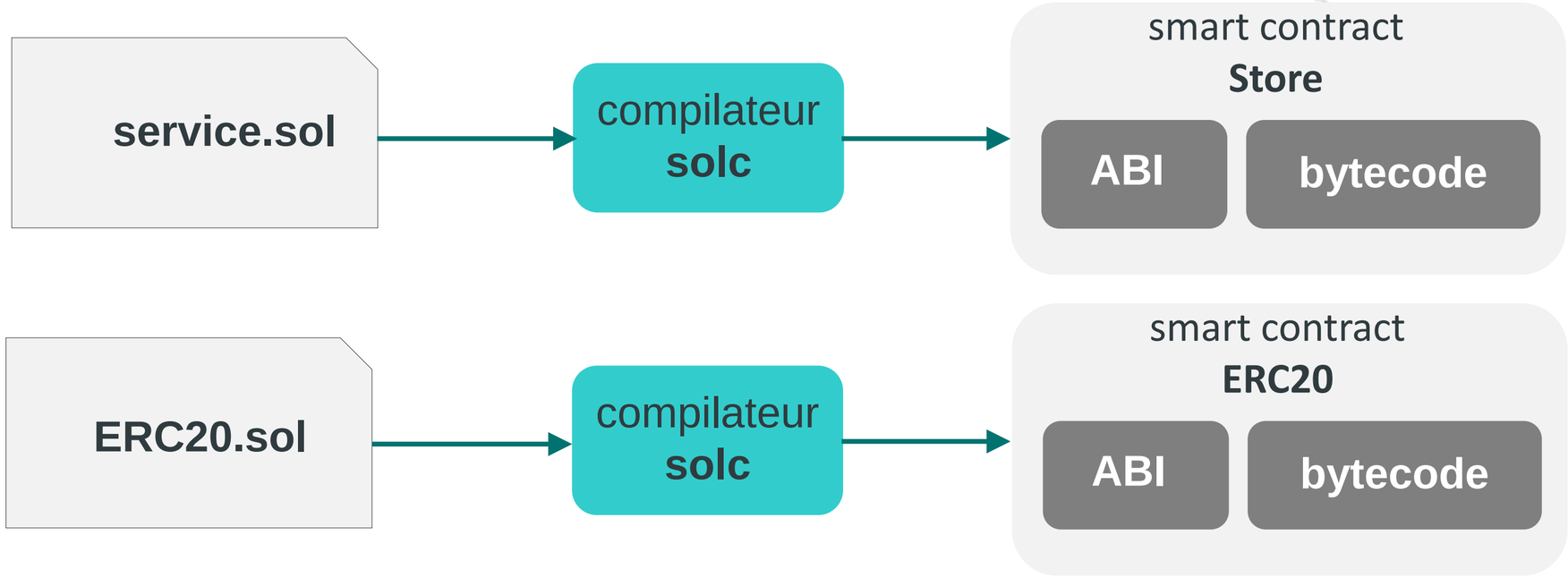
**Smart Contract Weakness Classification and Test Cases**

https://swcregistry.io/

**Exemple**

```
mapping (address => uint256) private _balances;

function _transfer(address sender, address recipient, uint256 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



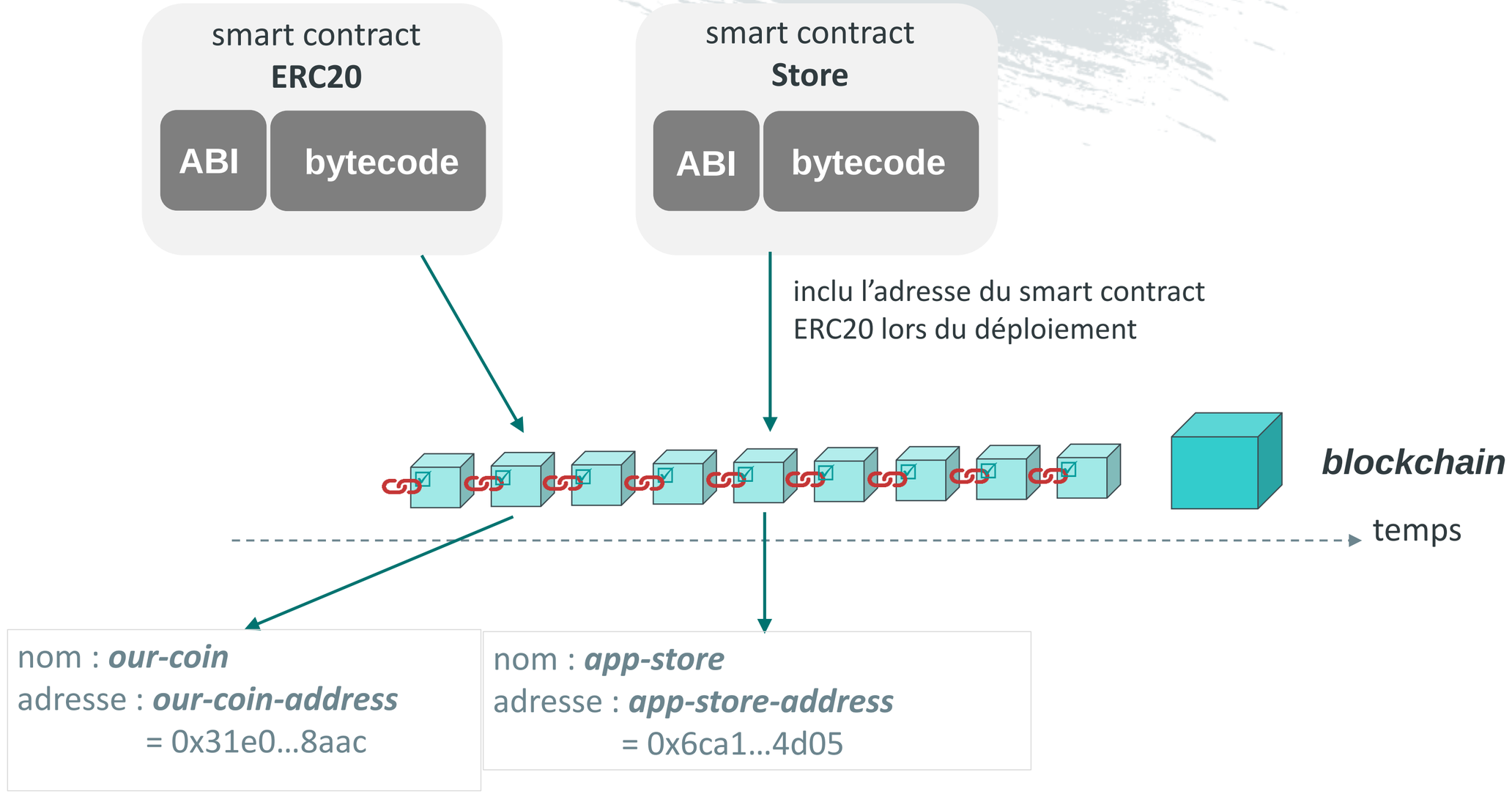
# COMPILATION 2/2

BasicToken = pyeth.contracts.contract('BasicToken',
b'6080604052348015610010576000800fd5b506104178061002060003960000f3006080604052600436106100575760003357c0100000000000000000000000000000000000000000000000000000000900463ffffffff16806318160ddd1461005c57806370a08231146100875780639059cbb146100de575b600080fd5b348015610068576000800fd5b506100716101435656b6040518082815260200191505060405180910390f35b348015610093576000800fd5b506100c860048036038101908080357ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff16906020019092919050505061014d565b6040518082815260200191505060405180910390f35b3480156100ea576000800fd5b506101296004803603810190808035773ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff16906020019092919050505061014d565b6040518082151515158152602001915050604051809103900f35b60006001549050565b60008060008373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002054905090919050565b60008073ffffffffffffffffffffffffffffffffffffffff168373ffffffffffffffffffffffffffffffffffffffff16141515156101d257600080fd5b6000803373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002054821115151561021f57600080fd5b6102708260008033373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002054610 3b490919063ffffffff16565b6000803373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002081905550610303826000808673ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002054610 3cd90919063ffffffff16565b6000808573ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002081905550827 3ffffffffffffffffffffffffffffffffffffffff1633 73ffffffffffffffffffffffffffffffffffffffff167fddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef8460405180828152602001915050604051809103 90a360019050929150505565b60008282111515156103c257fe5b8183039050929150505565b60008082840190508381101515156103e157fe5b8091505092915050565b600a165627a7a72305820cff2321f765033dd90f987a9d06d2bcd0cf0f6861d5c440616ec7815c440aeb10029',
[{"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":false,"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"_to","type":"address"},{"name":"_value","type":"uint256"}],"name":"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"}],"name":"Transfer","type":"event"}])

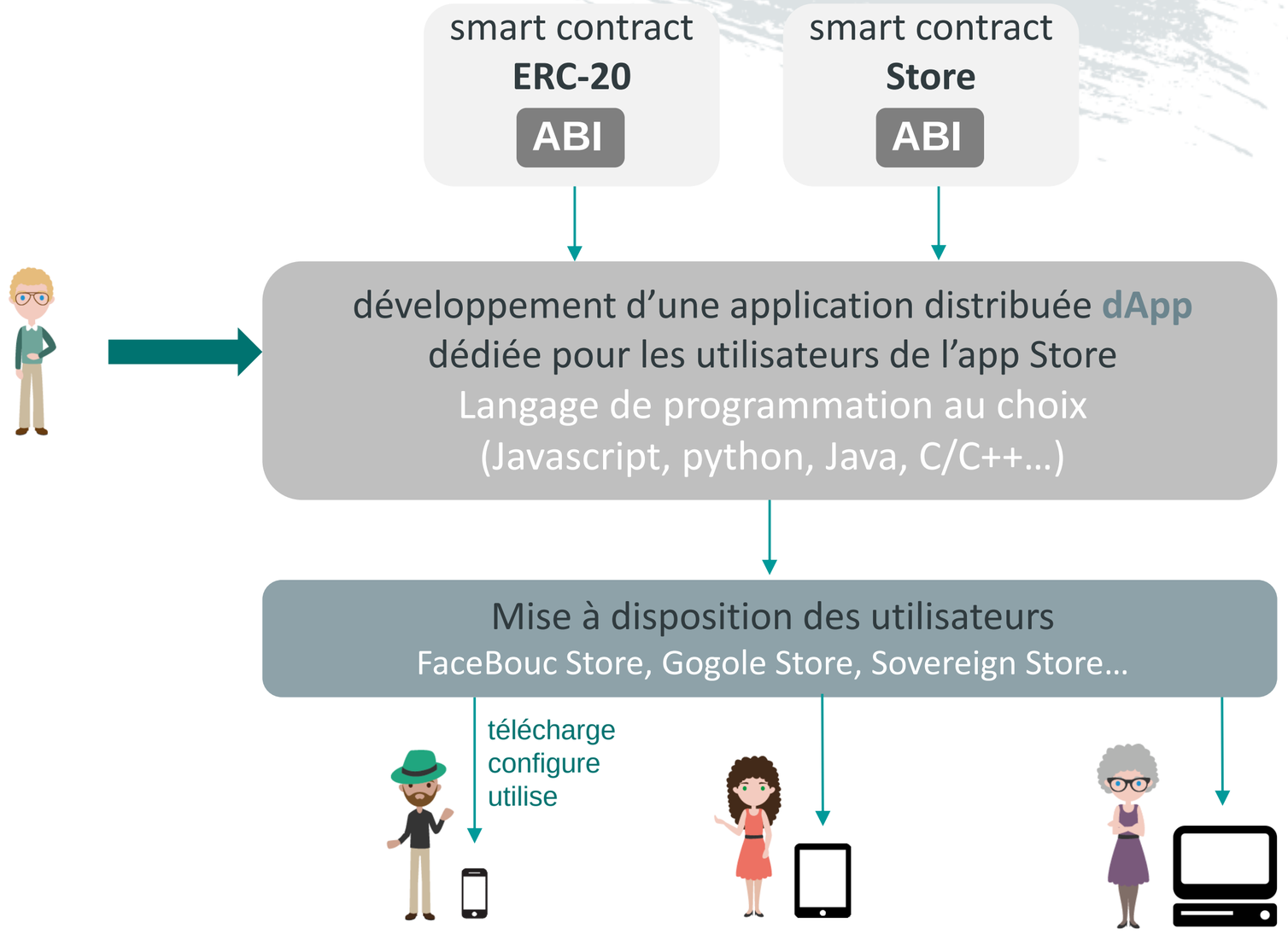
# BasicToken.sol

```
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];
                }
}
```

# DÉPLOIEMENT 1/2



# DÉPLOIEMENT 2/2



# QUELQUES RÉFÉRENCES


1. librairie openzeppelin, https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/token
2. librairie 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