FinTech R&D Innovation Conference

telindus

SnT securityandtrust.lu

Smart Contracts in the Ethereum platform.

Beltran Fiz , beltran.fiz@uni.lu
Radu State, radu.state@uni.lu

Sedan Lab, SnT

19th January 2016, Luxembourg

uni.lu Université du Luxembourg

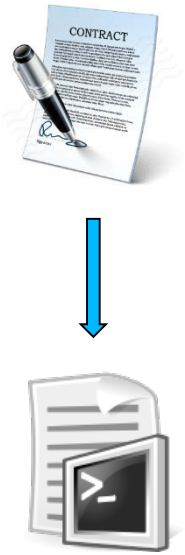
Smart contracts: What are they?

The term “smart contract” was coined by Nick Szabo in the 90s.

“self-automated computer programs that can carry out the terms of any contract.”

Ideally we would want this smart contract to be:

- Autonomous
- Transparent execution.
- Execution cannot be reverted
- Immutable
- Public



Trust in smart contracts

Web services today could already be considered “smart” contracts, however:

We trust that the central actor (Kickstarter):

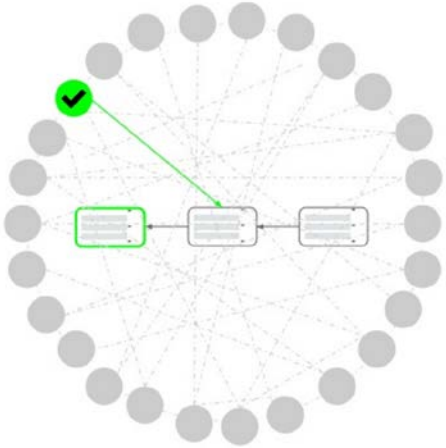
- Stores the funds safely (Custodial Trust)
- Acts fairly (House Trust)
- Keeps our private data safe (Data Trust)

A smart contract should not require any trust between its participants, and should be executed automatically following a set of defined rules.

Ethereum: What is it?

Ethereum builds upon the technology of bitcoin and uses some of the core features that made bitcoin a success such as:

- Underlying cryptocurrency
- Trustless peer to peer network.
- Inherent blockchain
- Decentralised concesus-based proof mechanism



Ethereum can be described as the evolution of the blockchain from a distributed transactional database to become a general purpose peer to peer computing framework

"Ethereum is to bitcoin what a smartphone is to a calculator" – Dr. Gavin Wood

Ethereum Virtual Machine (EVM)

The EVM is a decentralized computer containing millions of objects, called "accounts", which have the ability to:

- maintain an internal database,
- execute code using a Turing complete language
- communicate with other accounts

0x04: DIV	0x20: SHA3	0x45: GASLIMIT
0x05: SDIV	0x30: ADDRESS	0x50: POP
0x06: MOD	0x31: BALANCE	0x51: MLOAD
0x07: SMOD	0x32: ORIGIN	0x52: MSTORE
0x08: ADDMOD	0x33: CALLER	0x53: MSTORE8
0x09: MULMOD	0x34: CALLVALUE	0x54: SLOAD
0x0A: EXP	0x35: CALLDATALOAD	0x55: SSTORE
0x0B: SIGNEDEXTEND	0x36: CALLDATASIZE	0x56: JUMP
0x10: LT	0x37: CALLDATACOPY	0x57: JUMPI
0x11: GT	0x38: CODESIZE	0x58: PC
0x12: SLT	0x39: CODECOPY	0x59: MSIZE
0x13: SGT	0x3A: GASPRICE	0x5A: GAS
0x14: EQ	0x3B: EXTCODESIZE	0x5B: JUMPDEST
0x15: ISZERO	0x3C: EXTCODECOPY	0x6X: PUSHX
0x16: AND	0x40: BLOCKHASH	0xF0: CREATE

Ethereum Accounts

securityandtrust.lu

There are two types of addresses/accounts in Ethereum:

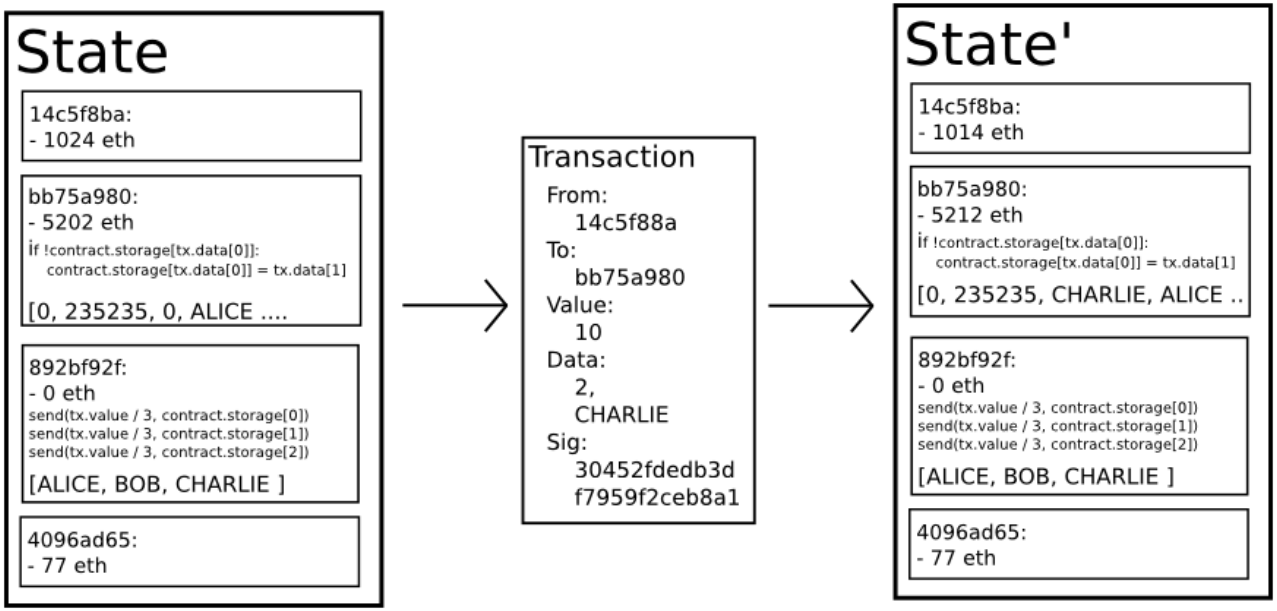
- External Accounts are controlled by a user who has an associated private key.
- Contracts Accounts are autonomous account which in addition to a balance they store the contracts code.
  - When a transaction is sent to a contract account, the account is executed
  - A contract can call other contracts

Therefore the current state of the EVM is kept in the blockchain and transactions are state transition functions.

UNIVERSITÉ DU LUXEMBOURG

Ethereum State Transition

Let’s go over an example state transition in Ethereum:



Halting Problem

Imagine a contract with an infinite loop execution and a transaction sent to this contract:

- Miners would attempt to validate the transaction
- Would stall the validation process.
- Basically an Ethereum denial of service attack.

Deciding if a program will terminate is undecidable over Turing complete languages.

Solution: Make each line execution consume payment tokens (Gas)

- Gas can be bought with Ether
- Each opcode consumes gas
- When you call a contract you provide gas.
- If gas runs out, execution aborts.
- Results are only stored if entire contract execution is completed.

Writing Contracts in Ethereum

SnT
securityandtrust.lu

ATTENTION

Frontier is an early access to the Ethereum network. Bugs and security issues might be present. Before downloading you have to agree to the terms that follow and the full terms linked below.

uni.lu
UNIVERSITÉ DU LUXEMBOURG

Writing Contracts in Ethereum

securityandtrust.lu

Contracts are stored on the blockchain in an Ethereum-specific binary format called EtherScript (Ethereum Virtual Machine bytecode).

```
TXVALUE PUSH 25 PUSH 10 PUSH 18 EXP MUL
LE NOT PUSH 34 JMPI STOP PUSH 34 JMP
PUSH 0 TXDATA SLOAD NOT PUSH 0 TXDATA
PUSH 1000 LT NOT MUL NOT NOT PUSH 34
JMPI STOP PUSH 1 TXDATA PUSH 0 TXDATA
SSTORE
```

Contracts tend to be written in a high level language which is then compiled into bytecode.

- Mutan (C like language)
- LLL (Lisp like language)
- Serpent (Python like language)
- Solidity (Javascript like language)

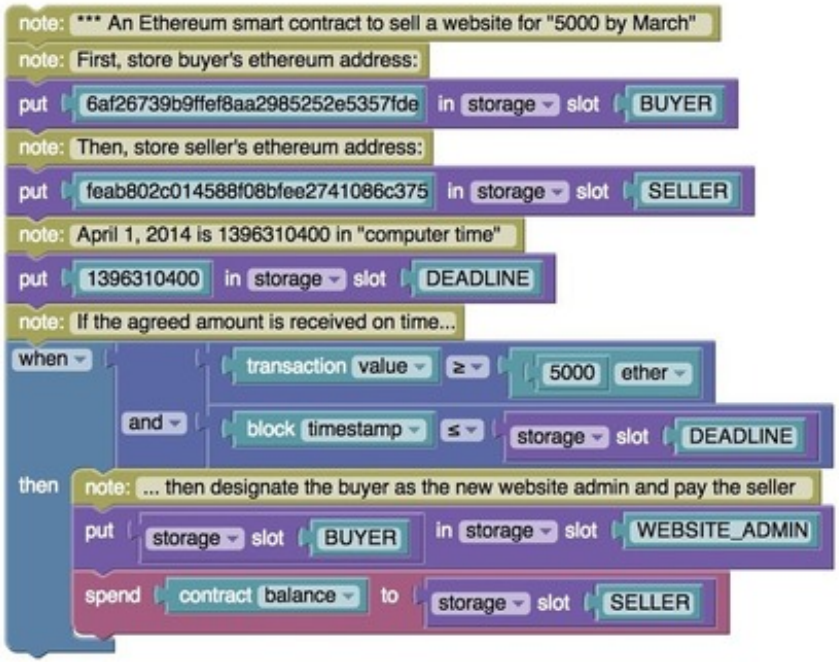
Solidity is however the most maintained language and is therefore recommended.

UNIVERSITÉ DU LUXEMBOURG

Writing Contracts in Ethereum

securityandtrust.lu

There are also more simple methods for contract creation being developed, such as Etherscripter:



UNIVERSITÉ DU LUXEMBOURG

Steps to deploy a contract

The current process for deploying a contract can be split into the following steps:

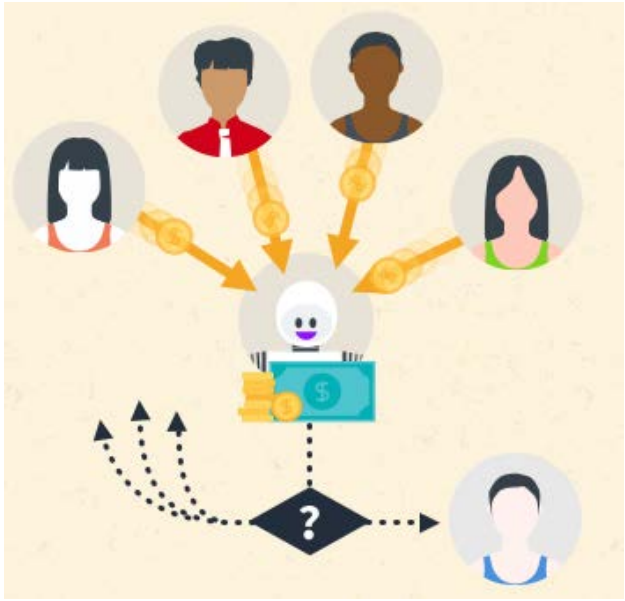
1. Define the contracts business logic
2. Write the code for the contract
3. Compile and test the contract locally
4. Deploy contract in live network and verify existence in blockchain.

Let’s use the example from the beginning: A crowdfunding contract.

Step 1: Define business logic

Lets define our crowdsourcing program:

- The project owner will create a contract account for each project .
- Each project has a funding goal and a deadline.
- This contract will create “backer tokens”
- When a backer sends Ether to the contract, he will receive backer tokens.
- When the deadline is over if the goal is reached, we send the ether to the creator.
- If funding goal is not reached, we send it back to the backers.
- We will allow backers to keep the tokens even if funding goal is not reached (proves they helped)



Step 2: Code the contract

securityandtrust.lu

Here is what the contract written in solidity looks like:

```
contract Crowdsale {

    address public beneficiary;
    uint public fundingGoal; uint public amountRaised;
    uint public deadline; uint public price;
    token public tokenReward;
    Funder[] public funders;
    event FundTransfer(address backer, uint amount, bool isContribution);

    /* data structure to hold information about campaign contributors */
    struct Funder {
        address addr;
        uint amount;
    }

    /*  at initialization, setup the owner */
    function Crowdsale(address _beneficiary,
    uint _fundingGoal, uint _duration, uint _price, token _reward) {
        beneficiary = _beneficiary;
        fundingGoal = _fundingGoal;
        deadline = now + _duration * 1 minutes;
        price = _price;
        tokenReward = token(_reward);
    }

    /* Default functio, called whenever anyone sends funds to a contract */
    function () {
        uint amount = msg.value;
        funders[funders.length++] = Funder({addr: msg.sender, amount: amount});
        amountRaised += amount;
        tokenReward.sendCoin(msg.sender, amount / price);
        FundTransfer(msg.sender, amount, true);
    }

    modifier afterDeadline() { if (now >= deadline) _ }

    /* checks if goal or time limit has been reached and end campaign */
    function checkGoalReached() afterDeadline {
        if (amountRaised >= fundingGoal){
            beneficiary.send(amountRaised);
            FundTransfer(beneficiary, amountRaised, false);
        } else {
            FundTransfer(0, 11, false);
            for (uint i = 0; i < funders.length; ++i) {
              funders[i].addr.send(funders[i].amount);
              FundTransfer(funders[i].addr, funders[i].amount, false);
            }
        }
        suicide(beneficiary);
    }
}
```

For a more detailed walkthrough, visit the Ethereum website.

uni.lu
UNIVERSITÉ DU LUXEMBOURG

Step 3: compile and test the contract

securityandtrust.lu

In order to test the contract it is higly recommended to first deploy the contract in a local ethereum network using a private blockchain.

A battery of tests should be performed to ensure exceptions are adequately handled.

We can then compile the contract using an ethereum node or an online compiler such a the one available at https://chriseth.github.io/browser-solidity/

The following byte code shows how the compiled code is stored in the blockchain:


```
6060604052600060016000505560405160208061077a833981016040528080519060200190919050505b33600360006101000a81548173ffffffffffffffffffffffffffffffffffffffff021916908302179055505b5061071
6806100646000396000f3606060405236156100745760003576c0100000000000000000000000000000000000000000000000000000000900480633c1d3491461007657806638da5cb5b146101415780
639003adfe1461017a578063a480ca791461019d578063a60f3588146101b5578063d4734f4a146101d857610074565b005b61008c60048083590602001909190505061022e565b6040518080602001
8373ffffffffffffffffffffffffffffffffffffffff168152602001828103825284818154600181600116156101000203166002900481526020019150805460018160011615610100020316600290048015610131578060f10
6101065761010080835404028352916020019161013156 5b8201919060005260206000020905b8154815290600101906020018083116101145782900360 1f168201915b505093505050506040518091 0
390f35b61014e60048050506102965 65b604051808273ffffffffffffffffffffffffffffffffffffffff168152602001915050604051809103 90f35b6101876004805050610 28d565b604051808281526020019150506040518
0910390f35b6101b360048080359060200190919050506106655 65b005b6101c260048050506102845 65b604051808281526020019150506040518091039 0f35b61022c60048080359060200190820 18
03590602001919190808060 1f0160208091040260200160405190810160405280939291908181526020018383808284378201915050505050509090919050506102bc565b005b600600050818154 81
10156100025790600052602060002090600202016000 5b9150905080600001600050908060010160009054906101000a900473ffffffffffffffffffffffffffffffffffffffff16905082565b600160005054815 65b6002600
0505481565b600360009054906101000a900473ffffffffffffffffffffffffffffffffffffffff1681565b60006000670de0b6
```


UNIVERSITÉ DU LUXEMBOURG

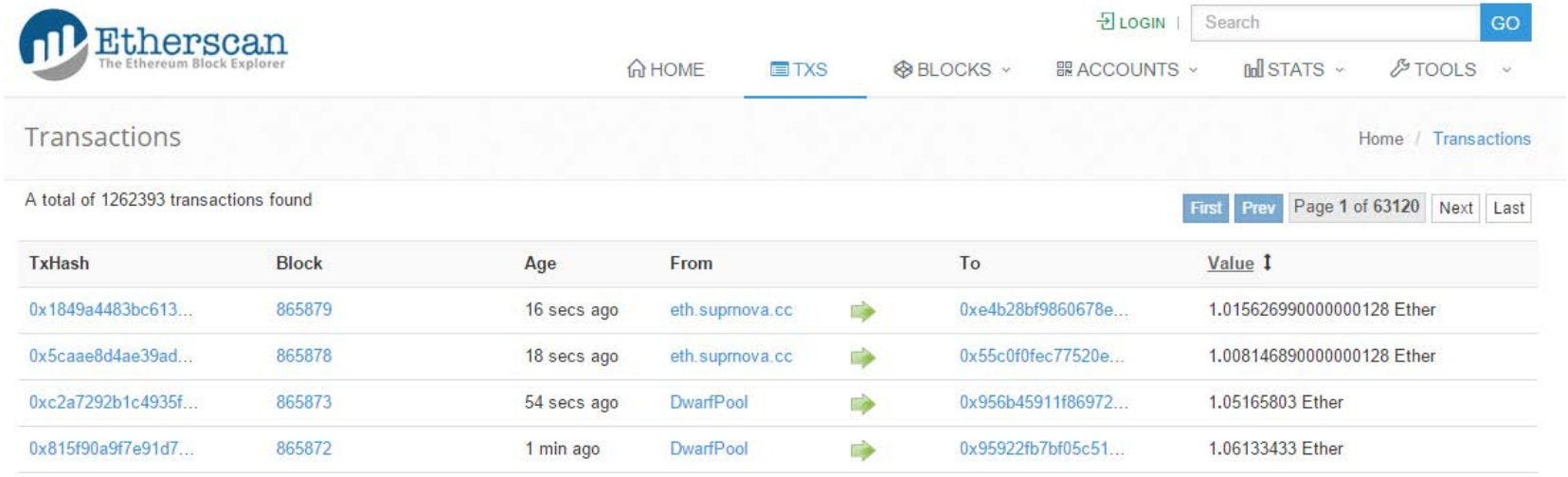
Step 4: deploy and check

securityandtrust.lu

Once deployed, we can ping our contract in the network to check it has been mined using the command:

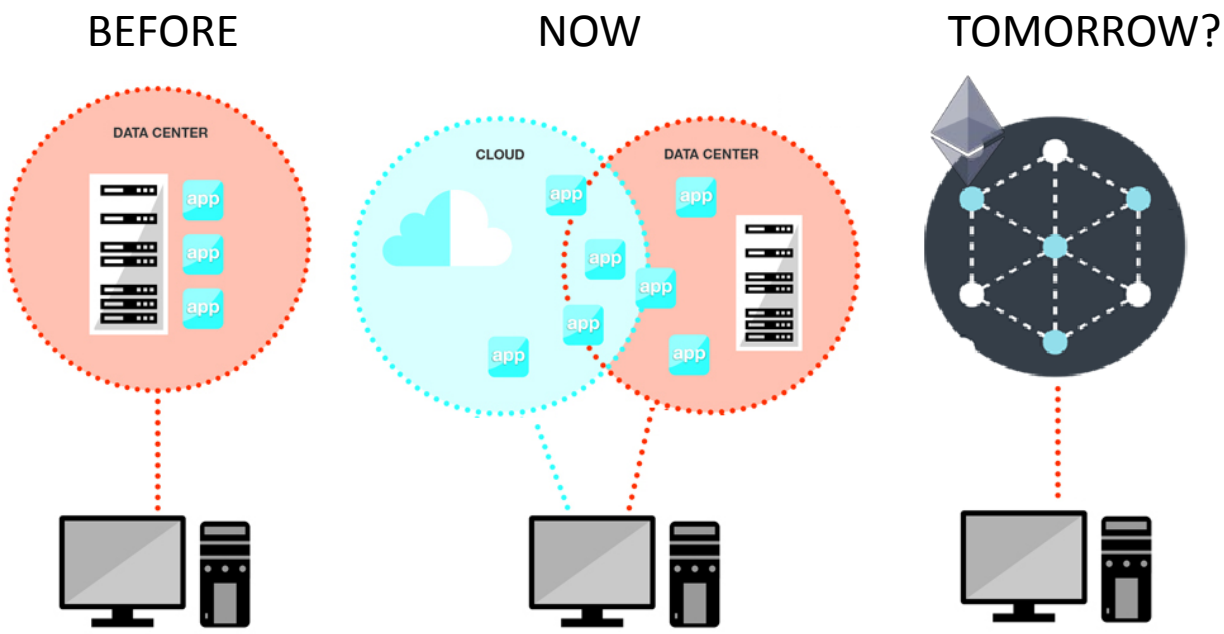
eth.getCode(codeAdddress)

Or we can use a blockchain explorer such as Etherscan.io :



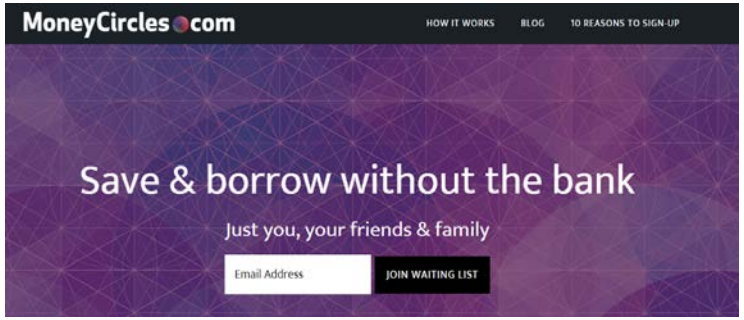
UNIVERSITÉ DU LUXEMBOURG

Towards decentralized apps



Some Decentralised Apps

Augur:



MoneyCircles

Slock.it:


I THINK WE SHOULD BUILD A BLOCKCHAIN
UH-OH
DOES HE UNDERSTAND WHAT HE SAID OR IS IT SOMETHING HE SAW IN A TRADE MAGAZINE AD?
WHAT COLOR DO YOU WANT THAT BLOCKCHAIN?
I THINK MAUVE HAS THE MOST RAM.
S. Adams
E-mail: SCOTTADAMS@AOL.COM
© 1995 United Feature Syndicate, Inc.(NYC)

Thank you