P2P Systems and Blockchains
Spring 2018,
instructor: Laura Ricci
laura.ricci@unipi.it

Lesson 18:
SMART CONTRACTS
EDITING AND DEPLOYMENT
16/05/2018
This lesson will present a step-by-step simple procedure for:

- editing smart contracts
- deploying smart contracts
- interacting with smart contracts on different Ethereum (test) networks

Environments:

- **MetaMask**
  - an Ethereum wallet implemented as a plugin (available for Google Chrome, Firefox,...)
- **Remix**
  - an in-browser Solidity editor to edit and test contract code
- **Public test network**
  - Ropsten network
- **Etherscan**
  - Ethereum blockchain explorer
“MetaMask is a bridge that allows you to visit the distributed web of tomorrow in your browser today. It allows you to run Ethereum DAPPS right in your browser without running a full Ethereum node.”

- simple to use wallet
- acts like a browser extension
  - easy to install
- supports various networks
  - main Ethereum
  - testnets like Ropsten
  - local testnet
- https://metamask.io/
• install the MetaMask plugin in your Google Chrome Browser
• create an account with a password
• if you click on an icon of Metamask in your Google Chrome Browser, and Login into your account
• you will see the window on the left
• you can buy or send Ether to a known address
ETHEREUM NETWORKS: ROPSTEN

- PoW testnet started in November 2016.
- named after a subway station in Stockholm.
- DoS attacked in February 2017 which made synching slow and made clients consume a lot of disk space. Revived in March 2017

- pros:
  - best reproduces the current production environment, i.e. system and network conditions on the live Ethereum main net, because it's PoW net.
  - can be used with both geth and parity.
  - Ether can be mined. Or requested from a faucet:
    - https://faucet.metamask.io/
    - http://faucet.ropsten.be:3001

- cons:
  - not immune to spam attacks
  - block time: 30 seconds
**ETHEREUM NETWORKS: RINKEBY**

- PoA (Proof of Authority) testnet started by the Ethereum team. Uses Clique PoA consensus protocol.
- started in April 2017.
- named after a metro station in Stockholm.
- pros:
  - immune to spam attacks (as Ether supply is controlled by trusted parties)
- cons:
  - supported by geth only
  - doesn't fully reproduce the current production environment as it uses PoA.
- Ether can't be mined. It has to be requested from the faucet:
  - https://faucet.rinkeby.io/
- block time: 15 seconds
ETHEREUM NETWORKS: KOVAN

- PoA testnet started by the Parity team
- started in March 2017.
- named after a subway station in Singapore.
- pros:
  - immune to spam attacks (as Ether supply is controlled by trusted parties)
- cons:
  - not supported in geth, supported by parity
  - doesn't fully reproduce the current production environment as it uses PoA.
  - Ether can't be mined. It has to be requested from the faucet.
  - block time: 4 seconds
ACCOUNTS AND ETHERSCAN

- You can browse your account on Etherscan
HOW TO GET ETHER?

- use the testnet faucet, http://faucet.ropsten.be:3001/

Ethereum Ropsten Faucet.

Enter your testnet account address
0xffc8298127770b0d6173a60859bb465daa6db46c

Send me 1 test ether!

This faucet drips 1 Ether every 7 seconds. You can register your account in our queue. Max queue size is currently 5. Serving from account 0x687422eaa2cb73b5d3e242ba5456b7e2919af:85 (balance 2901417 ETH)
Example command line: wget http://faucet.ropsten.be:3001/donate/<your ethereum address>
Example REST API: http://faucet.ropsten.be:3001/donate/<your ethereum address> API docs

Faucet queue
The queue is empty
SENDING AND RECEIVING TRANSACTIONS

- MetaMask may be exploited to send and receive transactions
  - receiving a transaction: the balance indicator in MetaMask increases.
  - sending transactions to other EOA or smart contracts account: simply press the SEND button.
    - once a transaction is sent, you can view it on Etherscan.
- You will also be able to find transactions associated with your address.
REMIX

- An in-browser Solidity editor to edit, test contract code and deploy them on different blockchains
- Full IDE to build/debug contracts
- https://remix.ethereum.org
- Functionalities: Development of smart contracts
  - Solidity editor
  - debug a smart contract’s execution.
  - access the state and properties of already deployed smart contract.
  - debug already committed transaction.
  - analyze solidity code to reduce coding mistakes
  - together with any tool which inject web3, Remix can be used to test and debug a contract on a real blockchain
pragma solidity ^0.4.0;

contract TestContract {

    struct Proposal {
        uint voteCount;
        string description;
    }

    address public owner;
    Proposal[ ] public proposals;

    function TestContract() {
        owner = msg.sender;
    }
}
function getowner () constant returns(address){
    return owner;
}

function setowner (address newowner){
    owner = newowner;
}

function createProposal (string description) {
    Proposal memory p;
    p.description = description;
    proposals.push(p);
}

function vote(uint proposal) {
    proposals[proposal].voteCount += 1;
}
REMIX: A WEB BASED IDE FOR SOLIDITY

- URL: remix.ethereum.org

- structured in three different zones:
  - **left**: file editor (presents an already inserted contract ballot.sol)
    - ability of creating a new contract in the browser
  - **central**: editor
    - insert code here
  - **right**: tools for the contact analysis and deployment
    - compile, debug, run,....

```
Sololidity ^0.4.0;

ballot.sol
pippo.sol

config
```
**REMIX: A WEB BASED IDE FOR SOLIDITY**

- **run tab (right part)**: for the contract deployment
- **different deployment environments**
  - **JavaScript VM**: a testnet which is run by the Remix IDE itself
  - **injected web3**: to deploy through MetaMask on one of the Ethereum testnets
  - **web3 provider**: to connect to a local testnet

<table>
<thead>
<tr>
<th>Environment</th>
<th>Account</th>
<th>Gas limit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript VM</td>
<td>Injected Web3</td>
<td></td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>Web3 Provider</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>5000000000000000000000</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Deploy**

- **Load contract from Address**
- **At Address**

**Pending transactions**

**Contract instances**
Deploying on the JavaScript VM.

- choose
  - an account for the contract from a predefined list
  - a gas limit
  - value: amount to be sent to this contract
  - the name of the contract to deploy: more contracts can be contained in the same file
- click on the deploy button
  - Before making the deploy, compile the contract
pragma solidity ^0.4.0;

contract pippo

creation of pippo pending...

[vm] from: 0xca3...a733c to: pippo,(constructor) value: 0 wei
data: 0x608...e0029 logs: 0 hash: 0xc65...f325b

status: 0x1 Transaction mined and execution succeed
contractAddress: 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
from: 0xca35b7d915458ef540ade6068df2f44e8fa733c
to: pippo.(constructor)
gas: 3000000 gas
transaction cost: 68794 gas
execution cost: 10666 gas
input: 0x60806040523435600057560000f0d5b50603580601d600030,0000f0d5b50603580601d600030,0000f0d5b50603580601d600030f
30056080052500080fd00a165627a77a72305820fb7bff036519eb8421b7382049ae15df653494a6be3fc4be6ac40180c9aa36e0029
decoded input: {}
decoded output: -
logs: []
value: 0 wei

contract deployed at the address:

0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
### AFTER THE DEPLOYMENT

- **contract deployed at the address:** 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
- **blue buttons:** query the current values of the public read-only fields of this contract.
  - the “owner” corresponds to your address
  - the array of proposals is empty
### AFTER THE DEPLOYMENT

- see the content of the proposals array by clicking on the blue button
- now vote and look how voteCount increases
- look at the transactions in the bottom grey field, to look at what's going on.
• after the creation of a proposal, query the “proposals” field by index.
• click proposal “proposals” button with index 0
• the description should match what you typed in createProposals, and the voteCount should be 0.
Deploying a contract on a testnet

- switch the execution environment from JavaScriptVM to “Injected Web3”.

- this connects your Remix IDE to MetaMask, and therefore to the Ropsten public test network.

- go to MetaMask and submit the transaction
Deploying a contract on a testnet

- you will see that there is a pending transaction
- you can follow the life-cycle of your transaction on https://ropsten.etherscan.io/
MONITORING THE TRANSACTION

You can see that there is a pending transaction
- on the Remix interface
- in the Metamask icon in your browser
- on ropsten.etherscan
When the transaction is confirmed,

- event is notified in Remix
- methods can now be called and variables can be inspected.
MONITORING THE TRANSACTION

Transaction Information

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxHash</td>
<td>0x7d79ceff16917ad6024f096e0642466899597a006f20c6d70185452f06e18927</td>
</tr>
<tr>
<td>TxReceipt Status</td>
<td>Success</td>
</tr>
<tr>
<td>Block Height</td>
<td>3240612 (9 block confirmations)</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>1 min ago (May-15-2018 03:10:35 PM +UTC)</td>
</tr>
<tr>
<td>From</td>
<td>0xftc829812777060d5173e60859bb465daa6db46c</td>
</tr>
<tr>
<td>To</td>
<td>[Contract 0xc883df57780b61fc1d2279ae44c1fd6c9a266b Created]</td>
</tr>
<tr>
<td>Value</td>
<td>0 Ether ($0.00)</td>
</tr>
<tr>
<td>Gas Limit</td>
<td>457598</td>
</tr>
<tr>
<td>Gas Used By Tnx</td>
<td>457598</td>
</tr>
<tr>
<td>Gas Price</td>
<td>0.0000000001 Ether (1 Gwei)</td>
</tr>
<tr>
<td>Actual Tx Cost/Fee</td>
<td>0.000457596 Ether ($0.000000)</td>
</tr>
<tr>
<td>Nonce</td>
<td>4</td>
</tr>
<tr>
<td>Input Data</td>
<td></td>
</tr>
</tbody>
</table>
DEPLOYING A CONTRACT ON ROPSTEN

- it is possible to interact with the contract in Remix with the same procedure shown in the previous slides

- click the “createProposal” and the “vote” method buttons and view the public fields.

- each method invocation is a transaction:
  - each time you invoke a method, it will pass it through to MetaMask, which will pop up a window asking you to confirm the transaction
  - You will notice your MetaMask balance slowly decreases as you make transactions, since you are spending gas, and paying for it using testnet coins.

- try to call the createProposal method and then to check if the new proposal has been inserted by inspecting the Proposals field
  - It will take some time before the transaction including the createProposal is confirmed
CONCLUSION: GLOSSARY

- **Ropsten test network.**
  - the public test net of Ethereum. (There are alternatives, this is the currently best-supported one).

- **MetaMask**: https://metamask.io/
  - a wallet for Ethereum you can install as a Firefox plugin or a Google Chrome plugin.
  - This will hold your private key, and you can use it to approve transactions. It connects to a third-party service as a bridge to a test network peer to peer network.

- **Etherscan Block explorer**: https://ropsten.etherscan.io/
  - Shows you the status of transactions and contracts on the ropsten blockchain.
  - You can use its “Verified and Publish” feature to register the source code for a contract you publish.
CONCLUSION: GLOSSARY

- **Remix IDE** [https://remix.ethereum.org/](https://remix.ethereum.org/)
  - An in-browser Solidity editor that lets you edit and test contract code.
  - It can also interoperate with MetaMask to interact with contracts on the testnet network.
  - It can generate transactions, which you can then approve and send through MetaMask.

- **Ropsten faucet**
  - A website that gives you free coins on the test network.
OTHER DEPLOYMENT ENVIRONMENTS

Truffle: https://github.com/trufflesuite/truffle

- Compile, deploy and test smart contracts
- Truffle contracts: a library of node.js
- https://github.com/trufflesuite/truffle-contract

JAVA + external libraries for interacting with WEB3
• Ethereum has opened a new world of applications beyond Bitcoin

• Smart Contracts makes the blockchain programmable

• Which applications?

• Work in progress...

• Let us give a first look to the possible applications and focus on one of them
Blockchain 1.0: currency, Bitcoin, Monero

Blockchain 2.0:
- Introduction of smart contracts to realize decentralized applications, Ethereum
- Further financial applications

Blockchain 3.0:
- Non cryptocurrencies related distributed ledgers
  - Electronic voting
  - Health care
  - Identity management systems
  - Decentralized notary
  - Supply chains Management
  - IoT

In the next slides we will focus on an application integrating IoT and supply chain and exploiting a distributed ledger
A SUPPLY CHAIN SCENARIO

- Alpha corporation designs and oversees the manufacture of complex multi-part equipment, for example for heavy industry, A1 is Alpha's head office.

- The equipment's components are produced and assembled in one of Alpha's factories, A2.
A SUPPLY CHAIN SCENARIO

- the equipment is shipped to different remote locations and used heavily. Regular servicing and maintenance is required to comply with local safety rules and legislation.

- Alpha contracts out equipment maintenance to authorized third parties, who use certified service engineers and approved replacement parts
A SUPPLY CHAIN SCENARIO

The scenario

- equipment may be sold from one corporation to another, and a record of the service history and provenance is vital
- at the end of the life-cycle the equipment is decommissioned
A SUPPLY CHAIN SCENARIO

- A1 runs a **centralized database** to track
  - Components
  - Equipments
  - Locations
  - service histories and life-cycle.

- a denial of service attack on a database (or even database server farm) may result in a serious risk
  - records in any databases can be altered or deleted

- some processes such as replacement part orders are manual, requiring human memory and intervention.
A SUPPLY CHAIN SCENARIO

• sharing access to the databases may be limited or be blocked by participants at any time without notice.

• problems to be faced:
  • compliance and third-party auditing
  • access permission granting and revocation
  • interoperability between different systems
  • disputes can be difficult to solve at a later date
  • costly and protracted legal action.
A SUPPLY CHAIN EXPLOITING BLOCKCHAIN

Component tracking and servicing on a blockchain has several advantages

- **blockchains are distributed:** different nodes in different locations run the software powering the blockchain and keep a copy of the data. More nodes implies:
  - more backup copies of the data
  - more backup “servers”
  - denial of service attacks become impossible

- **blockchains are tamper-proof:** as soon as a record is inserted into a blockchain, it is locked down, and cannot be changed.
  - hacking the data records becomes impossible
  - “delete and deny” defenses become impossible
  - auditing: a simple scan over the blockchain
component tracking and servicing on a blockchain has several advantages:

- **blockchains can support “smart contracts”**
  - components have their required service and replacement history pre-loaded onto the blockchain using a “smart contract”
  - when the scheduled service date or wear-and-tear usage limit is reached, the system automatically triggers a service request
  - on completion of the service or replacement, a record is generated on the blockchain
A SUPPLY CHAIN EXPLOITING BLOCKCHAIN

- the Alpha company:
  - starts a “permissioned blockchain” with a “genesis block”
  - runs the first block on the chain on a computer in its head office

- the genesis block contains the announcement message of Alpha’s first public key
  - this key will identify Alpha head office on the blockchain in future.

- the blockchain runs, and at regular intervals new blocks are added by Alpha’s blockchain node.
A SUPPLY CHAIN EXPLOITING BLOCKCHAIN

- Alpha’s factory A2 and the contracted service company B
  - generate public/private key pairs
  - send the public keys to A1, who announces it on the blockchain
- now A2 and B join the blockchain
  - run their own blockchain nodes
  - add blocks onto the blockchain
  - are nodes in the blockchain P2P network
- A1 does not know A2 and B’s associated private keys and they don’t A1’s private key
  - no one of the participants can impersonate each other
  - a message on the blockchain signed with B’s private key and verified with its public key is verified to come from B.
A SUPPLY CHAIN EXPLOITING BLOCKCHAIN

- A2 which manufactures components
  - creates a unique private/public key pair for each component
  - announces the public key on the blockchain, along with the location of the part, which is its factory warehouse.
- the components are assembled to produce the final machine
  - another key pair is generated for the finished product and is also reported
cheap components: marked with a QR code of the public key for scanning,

- do not participate to the blockchain

- other RFID scanning devices read the key value and submit reports to the blockchain on their behalf.
more expensive components:

- marked with an active RFID tag with Bluetooth connectivity.
- can contact nearby network connected devices and submit reports to the blockchain,
• finished machinery may have
  • a fully connected IoT device, with good connectivity to the network
  • onboard GPS for positionin
  • possibly a lightweight blockchain node for participating in the generation of new blocks.

• an integrated RFID tag reader
  • scans the complete machine for all its components,
  • detects changes made to those components.
when a component is logged on the blockchain by its public key announcement, a “smart contract” can be attached to the announcement.

this script will be run by nodes maintaining the blockchain if certain conditions are met.

such contracts can trigger a service request, decommissioning, or part replacement order to automatically be sent to a service engineering company.

the complexity of the automation provided depends on the design of the blockchain system.
A condition of the smart contract is met, a node generating the current block sends out a service request to B.

B sends a service engineer to the machinery in order to replace the wornout part.

Service engineers are certified on the blockchain by their own private/public key, stored in their smartphone or their service tablet.

The engineer replaces the part, the tablet and the machine’s central IoT device may send reports back to the blockchain to record the event. The contract is met.
A SUPPLY CHAIN EXPLOITING BLOCKCHAINS

- D decides to revoke a service engineer’s certification: he/she lefts the company...

- D posts
  - a key revocation message on the block chain
  - signed with his/her private key, revoking key b555.

- D can do this because he/she issued and signed the key in the first place.

- all records made with key b555 prior to block 412 remain valid, however, and the records cannot be removed.
company X is the owner of the machine, and sells it to company Y.

- the record of ownership and the location change of the machine is recorded to the blockchain
- it provides a future immutable record of its provenance.

at the end of the life-cycle the equipment is decommissioned,

- this event is recorded on the blockchain, with a signature from the original manufacturer, A.
- this allows future trusted auditing of the application of environmental legislation to be conducted.
At the end of the process a complete record exists on the blockchain of all:

- participants
- components
- their locations and journeys, who replaced what parts and where and when
- transfer of ownership
- final decommissioning

The records cannot be altered or deleted afterwards

- individual participants can track progress during transit, and review the data after delivery, alteration or transfer
- only those directly concerned with or involved in the design, manufacture, repair or order fulfillment can create relevant records
- there is no central server or central point of failure
BLOCKCHAINS INITIATIVES IN SOME COUNTRIES

UK: Distributed Ledger Technology, Beyond Blockchain (UK Govt Science Advisor) – Dec 2015
Dubai: Dubai Claiming Position at Forefront of Blockchain Technology – Aug 2016
: Dubai Wants All Government Documents on Blockchain by 2020 - Oct 2016
Belarus: Researchers Propose Using Blockchain In E-Governance Of Belarus – Oct 2016
: Blockchain to Drive Wanxiang’s $30 Billion Smart Cities Initiative - Sep 2016
Singapore: Singapore’s Central Bank Pairs Up With R3 to Create Blockchain R&D Center – Nov 2016
: IBM is Opening a Blockchain R&D Innovation Centre in Singapore Jul 2016
India: Indian IT Consultancy TCS is Developing over “100” Blockchain Projects – Oct 2016
Russia: Russian Central Bank Sends First Distributed Ledger Transactions – Oct 2016
Thailand: Thailand’s KBank and IBM team on blockchain project – Nov 2016
Indonesia: This Emerging Tech Company Has Put Asia’s Tuna On The Blockchain – Sep 2016
INDONESIAN TUNA FISH SUPPLY CHAIN

- Indonesia is the largest tuna-producing country
  - ideal for assessing opportunities to increase transparency in fish and seafood supply chains.

- A WWF pilot project which use a combination of
  - RFID tags,
  - quick response (QR) code tags
  - scanning devices

  to collect information about the journey of a tuna at various points along the supply chain.

- a support to struggle against illegal and environmentally dubious fishing practices

- demonstrating true stewardship and full transparency.