IIC3272 — Criptomonedas y contratos inteligentes

Project 2: explanation and topics

1. Administrative stuff

You should form groups of three people each. Your group will select one of the projects detailed below, investigate the topic, write a short report on it, *and* give a 30 minute presentation on the topic. The presentations can be recorded and shared via dropbox/google drive, or we can agree on date/time and do a Zoom session if you prefer it that way.

Deadline to hand in your work: Wednesday, July 15th, 2020.

2. FAQ

Do we need to hand in the work we do?
Yes, you need to write a short report detailing your findings. If you did some implementation, a link to the repository containing it should be provided.

What should be the structure of my presentation?
You have 30 minutes, so there is sufficient time to quickly explain the setting and background, and then detail the specific problem that is tackled, and how is this resolved.

We have a size 9 font in our presentation so that all the information fits on the slides. Is this OK?
No! An important part of the evaluation is your ability to summarize the work you did in a short presentation. This is no easy task. In particular, a good short presentation requires better understanding of the topic than a bad long presentation. In this course you will need to be able to isolate the important details of your work and present it in an accessible way.

I don't like any of the proposed topics.
Propose a new topic. For this, send me an email (dvrgoc@ing.puc.cl).

We're stuck, we don't know how to proceed, or can not decide what to do next.
That's what I'm here for. Send an email, or arrange a consultation time! Also, some projects have a lot of work in them, so if you do not finish all of it you still might get the highest score if sufficient effort is put into it.

3. Projects

A representative of the group should send me an email ccing the other person in the group with two preferences for a project they would like to do. It is recommendable that you do this as soon as possible, in order to get the topic you wish to work on. One topic can not be taken by two groups.

Bellow follows a list of proposed projects.

3.1. PencaCoin

Using the implementation we developed in class, develop a small scale cryptocurrency. Assume that you only need to support P2PKH scripts, and that your nodes connect to each other synchronously (i.e. once established, the connection is there to stay). Define elements that need to be stored by each node (UTXO pool, live transactions pool, blockchain, etc.), create at least three different nodes that are not on the same computer (i.e. a node should be running on a computer of each group member). Bootstrap the system by launching some transactions, and mining them into blocks. To keep things simple, fine tune the mining difficulty so that you can get a new block in less than 2 minutes, and allow at most 5 transactions per block. Now connect a new node to your network and do an initial block download. Showcase how the system works.

3.2. Segwit

Study the segwit soft fork of Bitcoin and explain what it works. Furthermore, implement a segwit inside our Bitcoin implementation (well, the one we stole from Jimmy Song). You can start learning about segwit here:

- Programming Bitcoin: `https://github.com/jimmysong/programmingbitcoin/blob/master/ch13.asciidoc`
- BIP141: `https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki`

3.3. EtherCoin

Extend our implementation of Script to allow loops (while and for). Also, extend the execution model by defining the cost of executing each operation, and charge this cost from the transaction fee. When executing a script, if no money is left in the transaction fee, the evaluation should return an error. Design a mechanism that also leaves some money for the miner (say, a predetermined amount at least). Run some transactions through your system, and simulate what a miner would do when executing them. Given that there is no global state of the program, show how you can continue a single execution over a chain of different transactions. For this, define an infinite while loop, and show how one can continue a previous execution.

It is up to you to define they syntax and semantics of the new expressions, and see whether you can support more than a single variable. If needed, use an additional stack.

3.4. SPV wallet

Using the code we have implement and test a fully operational SPV wallet. In particular, you should be able to manage different P2PKH and P2SH addresses, make payments, and synchronize your wallet each time the user asks you to (in which case you need to have the headers blockchain up to date, and MerkleProofs of all of interesting transactions). Your synchronization should work by connecting to at least three different nodes. Test the implementation on testnet.

3.5. Smart contract applications

In this project you will explore Bitcoin as a platform for running other applications. You should explore coloured coins and other smart contract applications and explain how they work/how can they de implemented. Reading is a selection from Chapter 9 of the Bitcoin Book.

3.6. Ethereum Modified Merkle Patricia Trees

Ethereum uses a more efficient data structure than a Merkle tree to store the state data. In fact, each Ethereum node stores the entire global state. The secret how this is done are Modified Merkle Patricia Trees. In this project you are asked to study this data structure, explain how it can be implemented, and why it works so efficiently (in particular how the state gets updated from one block to another). Furthermore, you are asked to implement this data structure, and do some basic experimentation to show how it works. Good references to start exploring this are (remove all \ and \\ from the links):

- Non technical overview: `https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/`
- High level description: `https://easythereentropy.wordpress.com/2014/06/04/\\ understanding-the-ethereum-trie/`
- Formal(ish) specification: `https://github.com/ethereum/wiki/wiki/Patricia-Tree`
- A nice discussion: `https://ethereum.stackexchange.com/questions/6415/eli5-\ how-does-a-merkle-patricia-trie-tree-work`
- Ethereum Yellow Paper Appendix D: `https://ethereum.github.io/yellowpaper/ paper.pdf#appendix.D`

3.7. Ethereum mining

Study the proof of work system that Ethereum deploys, and what is the struture of the reward. You should review how this changed historically, and what it is at the current moment. You should also compare it with the original motivation which comes from the following paper proposing the GHOST mining protocol:

- The GHOST protocol: `https://www.avivz.net/pubs/15/btc_ghost_full.pdf`

3.8. Ethereum scripting language

Analyse the Ethereum scripting language, also denoted as Ethereum Virtual Machine, highlight how its logic and how it differs from Script which is used in Bitcoin. You should also implement a small fragment of the Ethereum scripting language and show how it can simulate some simple features/contracts. Start exploring the language at:

- Ethereum wiki`https://github.com/ethereum/wiki/wiki/Ethereum-Virtual-Machine-\`
  `\(EVM)-Awesome-List`

3.9. Solidity vulnerabilities

Select a real world example of a known Ethereum contract that was found to have a vulnerability. Analyse the code, and show what the vulnerability is. If possible, replicate the attack on a test network (either locally with truffle, or on a testnet). Links to several pages listing real world vulnerabilities can be found in the following resource (in every subsection called "Real world example"):

- Solidity vulnerabilities: `https://github.com/ethereumbook/ethereumbook/blob/develop/`
  `09smart-contracts-security.asciidoc`

3.10. Create your own user application for Ethereum

In this project you are required to propose a good use case scenario for implementing on the Ethereum blockchain. You should be very clear why it makes sense to implement this as a smart contract on Ethereum, and not simply as a distributed database or a centralized one. You should also implement your idea as a Solidity smart contract, deploy it, show how it works, and discuss why it is securely designed and does not have an obvious vulnerability.

3.11. Continue your work from Project 1

This should be self explanatory.

3.12. Propose your own topic

Quite simple: send me an email (dvrgoc@ing.puc.cl) to confirm the topic you would like to explore.