Implementing Cryptocurrency with Block Chains

Cryptography CS 411& CS 507 Term Project for Fall 2017

E. Savaş
Computer Science & Engineering
Sabancı University
İstanbul

Abstract

You are required to develop essential building blocks of cryptocurrency using block chains.

1 Introduction

The project has three phases:

- Developing software for proof of work
- Developing software for digital signature
- Developing software for other buiding blocks and integration

More information about the phases are given in the subsequent sections.

2 Developing software for proof of work

In cryptocurrency systems, miners approve transactions by running the proof-of-work (PoW) algorithm. A transaction contains information of a payment from the *payer* to the *payee* and is in the following format:

```
*** Bitcoin transaction ***
Serial number:
Payer:
Payee:
Amount:
Previous hash in the chain:
Nonce:
Proof of Work:
```

Explanations of these fields are as follows
`Serial Number:` is a uniformly randomly generated 128-bit integer;
`Payer:` is the identity of the person making the payment;
`Payee:` is the identity of the person receiving the payment;

`Amount:` is the amount in *Satoshi* being transferred;
`Previous hash in the chain:` is the hash of the last transaction in the chain
`Nonce:` is a uniformly randomly generated 128-bit integer, which miners change to find the PoW for transactions.
`Proof of Work:` is the hash of transactions whose first digits of certain length are all 0. This is written in hexadecimal format.

The following is an example of a block chain generated for three transactions:

```
*** Bitcoin transaction ***
Serial number: 212263538645046771822600700406703677780
Payer: Erkay Savas
Payee: 2KQMHBX7LR
Amount: 241 Satoshi
Previous hash in the chain: First transaction
Nonce: 29573528225660355634849978619601037 3709
Proof of Work: 00000042e244ebcdf74f456c35f674d18d014b1317c335beb717fb6af3012403
*** Bitcoin transaction ***
Serial number: 154066494402135221526580229746974924 47
Payer: Erkay Savas
Payee: JJCHLE0S8S
Amount: 353 Satoshi
Previous hash in the chain: 00000042e244ebcdf74f456c35f674d18d014b1317c335beb717fb6af3012403
Nonce: 22901188847043023983358186249537206075 8
Proof of Work: 00000039f0eb03ca50d6d7399717962e47f1189b70fc5b4abfea1c6df4436100
*** Bitcoin transaction ***
Serial number: 13466488668800169841482920553532463414 7
Payer: Erkay Savas
Payee: 8O41AGH4QM
Amount: 711 Satoshi
Previous hash in the chain: 00000039f0eb03ca50d6d7399717962e47f1189b70fc5b4abfea1c6df4436100
Nonce: 17404173832807533424784079016073135314 1
Proof of Work: 000000b48355d57cf79a79801ec96cbd83716b144a0d2896a9078a14b39fa202
```

As can be observed from this example, all PoWs start with six 0s. This can be achived by miners generating different `Nonce` values until one hash satisfying this condition is found. PoW is the hash of the first seven lines of a transaction excluding the last line. In the example we use `SHA3` with 256-bit output as the cryptographic hash function in python module `hashlib`.

In this phase of the project, you will generate random transactions following the format described above. You are required to use `SHA3` with 256-bit output as the cryptographic hash function. For the sake of simplicity, you need to provide a PoW for a single transcation as in the example in the file "LongestChain.txt" attached to the assignment package. You need to generate at least ten transactions and write them to a file.

You are required to submit your python source code(s) that you used to generate transactions and their PoWs along with a file that contains at least ten transactions. The transactions in your file should be in the format as the example transactions in the file "LongestChain.txt". Note that we will use the python code "ValidateChain.py" in the attachment to validate your transactions. Before you submit your work, you are advised to use it to check if your transactions are good.

For more information about cryptocurrencies see the attached document "bitcoin.pptx".

3 Developing software for digital signature

Details will be provided in the second phase of the project

4 Developing software for other buiding blocks and integration

Details will be provided in the third phase of the project

5 Appendix I: Timeline & Deliverables & Weight & Policies etc.

Project Phases	**Deliverables**	**Due Date**	**Weight**
Project announcement		01/12/2017	
First Phase	Source codes, Block chain file, Readme file	08/12/2017	30%
Second Phase	TBA	15/12/2017	40%
Third Phase	TBA	22/12/2017	30%

5.1 Policies

- You may work in groups of two (at most three)
- You may be asked to demonstrate a project phase to a TA or the instructor.
- In every phase, we will provide you with a validation software in python language that can be used to check your implementation for correctness. We will also use it to check your implementation. If your implementation in a project phase fails to pass the validation, you will get no credit for that phase.