CSCM29 - Blockchain, Cryptocurrencies and Smart Contracts

Arnold Beckmann, Anton Setzer
Lab 3

16 October 2019

Statement about Academic Integrity By submitting this coursework, electronically and/or hardcopy, you state that you fully understand and are complying with the university's policy on Academic Integrity and Academic Misconduct. The policy can be found at
`https://myuni.swansea.ac.uk/academic-life/academic-misconduct`

Introduction: In this lab the ledger based model introduced in the last lab is modified so that the users are replaced by public keys and transactions need to be signed by the private keys of those spending coins. Your task is to implement the logic used to process transactions and produce the ledger.

The repository located at `https://github.com/csetzer/CSCM29.git` has been updated to contain lab3. You can simply run in the directory which you have already downloaded in lab2

```
git pull
```

Should you get errors this might be because you have modified files. Then clone the repository from lab2 in a new directory.

You will be provided with the following classes:

- Input, Output defining single inputs and outputs to a transaction. Input have an extra field for the signature of a transaction
- InputList, OutputList, defining the list of inputs and outputs of a transaction.
- PublicKeyMap maintains a map between usernames (given as strings) and public key. It has both a map from names to public keys, and one from public keys to names. This map is mainly used for convenience, since is difficult to deal manually with public keys.
- PublicPrivateKeyMaps administrates in addition to what is in PublicKeyMap private keys. Only the originator of those keys should have access to it.
- SampleKeyMap allows to define a PublicPrivateKeyMap from a given list of user names.

- SigData administrates data to be signed. When defining transactions which compute the part of the transaction to be signed as an element of this class.
- The class Transaction defines transactions, and Ledger defines the ledger.
- Furthermore Crypto provides convenient methods for verifying signatures and singing messages.
- In LabUtils two more classes are defined which form utilities.

The tasks are

- **Task1:** Define in Transaction.java the method checkSignaturesValid() so that it checks whether the signatures in the transaction are correct. (At the moment this method always returns true).
- **Task 2** In Ledger.java define the method checkTransactionValid for checking that a transaction is valid (again currently it returns always true).
- **Task 3** Define the test cases in the method test() of Ledger.java using the same steps as before.

  In the part referring only to input and output lists you can use as signatures the random signedMessages defined in the code.

  Once you reach where one is dealing directly with transactions, you can define properly signed messages and check they are correctly signed.

  Create one example where an input is signed w.r.t. a different output list from the one being used. Checking whether it is correctly signed should then return false.