HW 3: cryptocurrency

Submit Assignment

Due Oct 16 by 11:59pm **Points** 20 **Submitting** a file upload **File Types** js

For this assignment, your job is to implement a cryptocurrency in JavaScript.

Following the peer-to-peer cryptocurrency model, there will be no central authority. Every client keeps a **ledger** that tracks how much money everyone has. Note that the public key for each user is hashed to serve as an anonymized ID.

When users wish to transfer money, they broadcast a 'transfer' message to the network. In order to prevent cheating, a 'proof' must be found for the transaction before users will update their ledgers. This proof is a random string that can be hashed with the ledger to produce the correct number of leading zeroes. When a 'miner' finds this proof, he/she broadcasts it to the network, receiving an extra, newly 'mined' coin as a reward.

Public & private keys are provided for three users:
*alicePriv.txt/alicePub.txt -- Private/public keys for Alice
*bobPriv.txt/bobPub.txt -- Private/public keys for Bob
*charliePriv.txt/charliePub.txt -- Private/public keys for Charlie

JavaScript starter code provided on the course website include:
*cryptoCurr.js -- The currency implementation. You will need to make your changes here.
*coinUI.js -- Command line interface for the GUI. Some sample usages:

```
  node coinUI.js alicePriv.txt alicePub.txt 9000
  node coinUI.js bobPriv.txt bobPub.txt 9001
```

*tester.js -- Some sample test cases
*testBadSig.js -- An attempt by Charlie to steal Alice's money.

The expected output for the test files is provided in tester.output and testBadSig.output. (Note that there might be some minor differences in the results due to the way that the protocol works).

Your tasks:

1) When a user transfers funds (the transferFunds method), sign the transaction details and add a 'sig' field to the transaction.

2) Update the validateTransfer method to verify the signature. The ledger should be updated only if the signature is valid, otherwise, broadcast a reject message. Use the 'broadcast' method provided. It takes an object with the message details. In this case, it should be:

```
    {type: 'reject', msg: 'bad signature'}
```

3) The validateTransfer creates a new ledger, but it is not official until a proof has been found, which is the responsibility of the mineProof method. Update mineProof to find a valid proof value.

4) Note that the design of CryptoCurr.js relies on event handlers, added through the CoinClient.prototype.on method. Add an additional handler for the event 'proof'. When it is received, verify that the proposed ledger is valid, according to these rules:

a. The length of the new ledger (specified by the chainLength method) must be longer than the client's current ledger.
b. The proof for the new ledger must be valid.

If the new ledger seems valid, the client should replace its current ledger with the new ledger.