

Blockchain

Wallets and Transactions on the Blockchain:

Wallets Keys and Transactions:

What is a wallet?

- Wallets store the balance of an individual.
- They store an individual's keys.

Private key

Used to generate signatures.

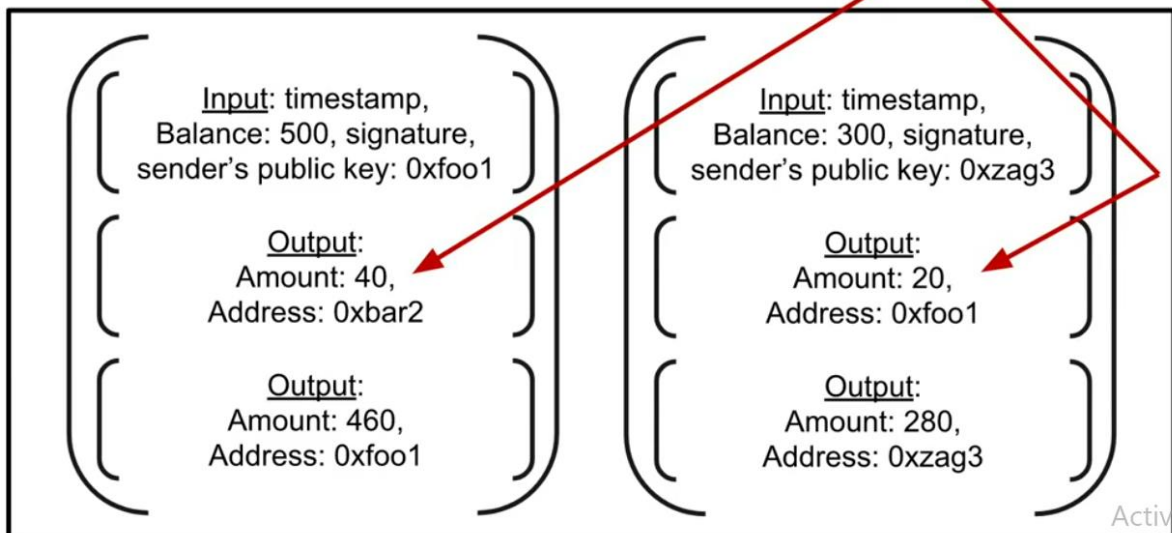
Public key

Used to verify signatures.

Also the public address.

Transactions

Balance: 0xfoo1



Blockchain

Digital Signatures



Blockchain-powered Cryptocurrencies

- Contain wallet objects.
- Keys for digital signatures and verification.
- Have transactions objects to represent currency exchange.

Blockchain

Create Wallet:

```
... JS index.js 1 X JS config.js
wallet > JS index.js > Wallet
1  const { INITIAL_BALANCE } = require('../config');
2
3
4  class Wallet {
5      constructor(){
6          this.balance = ;
7          this.keyPair = null;
8          this.publickey = null;
9      }
10
11     toString(){
12         return `Wallet -
13         publicKey: ${this.publickey.toString()}
14         balance : ${this.balance}`
15     }
16 }
17
18 module.exports = Wallet;
```

```
JS index.js 1 JS config.js X
JS config.js > <unknown> > INITIAL_BALANCE
1  const DIFFICULTY = 4;
2
3  const MINE_RATE = 3000;
4  const INITIAL_BALANCE = 500;
5
6  module.exports = { DIFFICULTY, MINE_RATE, INITIAL_BALANCE };
```

Blockchain

Chain Util and Key Generation

npm i elliptic --save

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm i elliptic --save
added 7 packages, and audited 377 packages in 2s

39 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

```
index.js > ...
1
2 const Wallet = require('./wallet');
3 const wallet = new Wallet();
4 console.log(wallet.toString());
5
```

```
wallet > JS index.js > Wallet
1 const ChainUtil = require('../chain-util');
2 const { INITIAL_BALANCE } = require('../config');
3
4
5 class Wallet {
6   constructor(){
7     this.balance = INITIAL_BALANCE;
8     this.keyPair = ChainUtil.genKeyPair;
9     this.publicKey = this.keyPair.getPublic().encode('hex');
10  }
11
12   toString(){
13     return `Wallet -
14     publicKey: ${this.publicKey.toString()}
15     balance : ${this.balance}`
16   }
17 }
18
19 module.exports = Wallet;
```

Blockchain

Create a Transaction

```
C:\Users\Dell\Desktop\BlockChain\Course II>npm i uuid --save

added 1 package, and audited 378 packages in 819ms

39 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

```
wallet > JS transactions.js > Transaction > constructor
1  const ChainUtil = require('../chain-util');
2
3  class Transaction {
4      constructor(){
5          this.id = ChainUtil.id();
6          this.input = null;
7          this.outputs = [];
8      }
9
10     static newTransaction(senderWallet, recipient, amount){
11         const transaction = new this();
12
13         if(amount > senderWallet.balance){
14             console.log(`Amount : ${amount} exceeds balance.`);
15             return;
16         }
17
18         transaction.outputs.push(...[
19             {amount: senderWallet.balance = amount, address: senderWallet.publicKey },
20             {amount, address: recipient}
21         ])
22
23         return transaction;
24     }
25 }
26
27
28
29 module.exports = Transaction;
```

Blockchain

Test the Transaction:

```
5 chain-util.js > ChainUtil > genKeyPair
1  const EC = require('elliptic').ec;
2  const uuidV1 = require('uuid/v1');
3  const ec = new EC('secp256k1');
4
5  class ChainUtil {
6    static genKeyPair(){
7      return ec.genKeyPair();
8    }
9
10   static id() {
11     return uuidV1();
12   }
13 }
14
15 module.exports = ChainUtil;
```

wallet > JS transaction.test.js > ...

```
1  const Transaction = require('./transactions');
2  const Wallet = require('./index');
3
4
5  describe('Transcation', () => {
6    let transaction, wallet, recipient, amount;
7
8    beforeEach(() => {
9      wallet = new Wallet();
10     amount = 50;
11     recipient = 'r3c1p13nt';
12     transactions = Transaction.newTransaction(wallet, recipient, amount);
13   });
14
15
16   it('outputs the `amount` subtracted from the wallet balance', () =>{
17     expect(Transaction.outputs.find(output => output.address === wallet.publicKey).amount)
18       .toEqual(wallet.balance - amount);
19   });
20
21
22   it('outputs the `amount` added to the recipient', () =>{
23     expect(transaction.outputs.find(output.address === recipient).amount)
24       .toEqual(amount);
25   });
26 })
```

{ } package.json > { } dependencies

```
12  "jest":{
13    "testEnvironment": "node"
14  },
```

Blockchain

wallet > JS transaction.test.js > describe('Transcation') callback > describe('transacting with an amount that exceeds the balance')

```
17
18   it('outputs the `amount` subtracted from the wallet balance', () =>{
19     expect(Transaction.outputs.find(output => output.address === wallet.publicKey).amount)
20       .toEqual(wallet.balance - amount);
21
22   });
23
24   it('outputs the `amount` added to the recipient', () =>{
25     expect(transaction.outputs.find(output.address === recipient).amount)
26       .toEqual(amount);
27   });
28
29   describe('transacting with an amount that exceeds the balance', ()=>{
30     beforeEach(() =>{
31       amount = 50000;
32       transaction = Transaction.newTransaction(wallet, recipient, amount);
33     });
34
35     it('does not create the transaction', () =>{
36       expect(transaction).toEqual(undefined);
37     });
38   });
39 })
```



Sign a Transaction:

Blockchain

JS chain-util.js > ChainUtil > hash

```
1  const EC = require('elliptic').ec;
2  const SHA256 = require('crypto-js/sha256');
3  const { v1: uuidV1 } = require('uuid');
4  //const uuidV1 = require('uuid/v1');
5
6
7  // const uuidV1 = require('uuid');
8
9  //   console.log(uuidV1.v1());
10
11
12  // var uuid = require('uuid');
13  // const uuidV1 = require('uuid/v1');
14
15  const ec = new EC('secp256k1');
16
17  class ChainUtil {
18      static genKeyPair(){
19          return ec.genKeyPair();
20      }
21
22      static id() {
23          return uuidV1();
24      }
25
26      static hash(data){
27          return SHA256(JSON.stringify(data)).toString();
28      }
29  }
```


Blockchain

wallet > JS index.js >  Wallet >  sign

```
1  const ChainUtil = require('../chain-util');
2  const { INITIAL_BALANCE } = require('../config');
3
4
5  class Wallet {
6      constructor(){
7          this.balance = INITIAL_BALANCE;
8          this.keyPair = ChainUtil.genKeyPair;
9          this.publicKey = this.keyPair.getPublic().encode('hex');
10     }
11
12     toString(){
13         return `Wallet -
14             publicKey: ${this.publicKey.toString()}
15             balance : ${this.balance}`
16     }
17
18     sign(dataHash){
19         return this.keyPair.sign(dataHash);
20     }
21 }
22
23 module.exports = Wallet;
24
25 //getPublic()
```

Blockchain

wallet > JS transactions.js > Transaction > newTransaction

```
16     }
17
18
19     transaction.outputs.push(...[
20       {amount: senderWallet.balance = amount, address: senderWallet.publicKey },
21       {amount, address: recipient}
22     ])
23
24     Transaction.signTransaction(transaction, senderWallet);
25
26     return transaction;
27
28   }
29
30   static signTransaction(transaction, senderWallet){
31     transaction.input = {
32       timestamp: Date.now(),
33       amount: senderWallet.balance,
34       address: senderWallet.publicKey,
35       signature: senderWallet.sign(ChainUtil.hash(transaction.outputs))
36     }
37   }
38 }
39
```

Test the Transaction Input:

wallet > JS transaction.test.js > describe('Transcation') callback > it('inputs the balance of the

```
28
29   it('inputs the balance of the wallet', ()=>{
30     expect(transaction.input.amount).toEqual(wallet.balance);
31   })
32
33
```

Verify Transactions:

JS chain-util.js > ChainUtil > verifySignature

```
29
30   static verifySignature(publicKey, signature, dataHash){
31     return ec.keyFromPublic(publicKey, 'hex').verify(dataHash, signature);
32   }
33 }
34
```

Blockchain

wallet > JS transactions.js > Transaction > verifyTransaction

```
39
40 static verifyTransaction any transaction){
41     return ChainUtil.verifySignature(
42         transaction.input.address,
43         transaction.input.signature,
44         ChainUtil.hash(transaction.outputs)
45     );
46 }
```

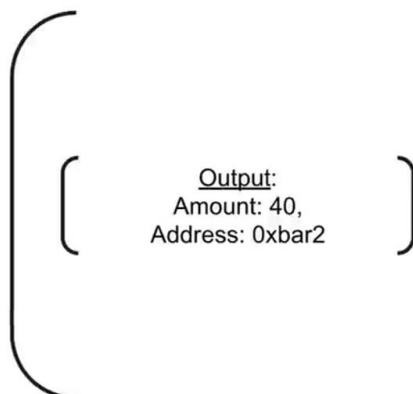
Test Transaction Verification:

wallet > JS transaction.test.js > describe('Transaction') callback

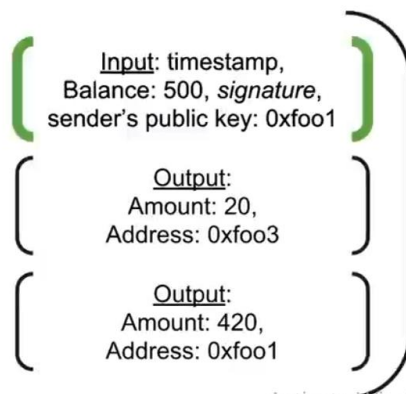
```
33
34 it('validates a valid transaction', () => {
35     expect(Transaction.verifyTransaction(transaction)).toBe(true);
36 });
37
38 it('invalidates a corrupt transaction', () => {
39     transaction.outputs[0].amount = 50000;
40     expect(Transaction.verifyTransaction(transaction)).toBe(false);
41 });
```

Transaction Updates:

Transaction Updates



Send 20 to 0xfoo3

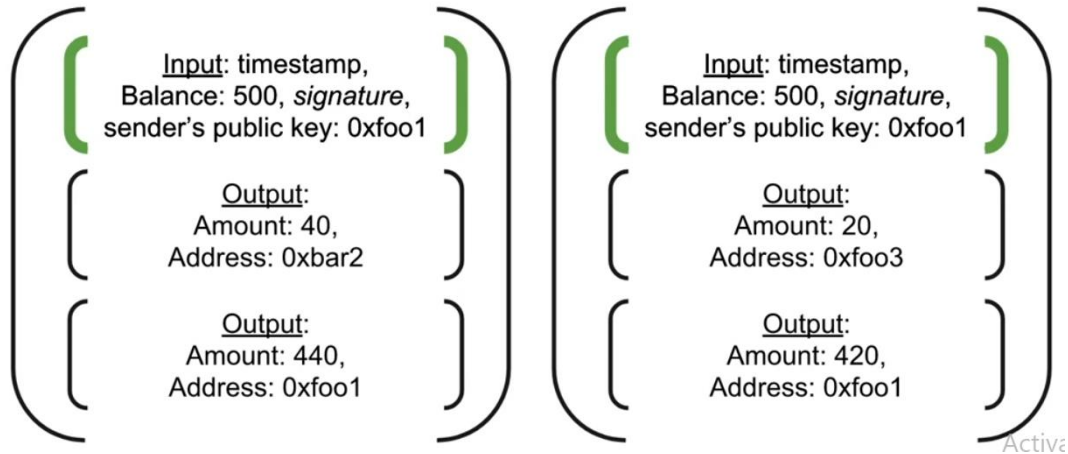


Aristotle Marada

Blockchain

Transaction Updates

Send 20 to 0xfoo3



wallet > JS transactions.js > Transaction > update

```
9
10 update(senderWallet, recipient, amount){
11     const senderOutput = this.outputs.find(output => output.address == senderWallet.publicKey);
12
13     if (amount > senderOutput.amount) {
14         console.log(`Amount : ${amount} exceeds balance.`);
15         return ;
16     }
17
18     senderOutput.amount = senderOutput.amount - amount;
19     this.outputs.push({amount, address: recipient});
20     Transaction.signTransaction(this, senderWallet);
21
22     return this;
23
24 }
```

Test Transaction Updates:

Blockchain

wallet > JS transaction.test.js > ...

```
54     describe('and updating a transaction', () =>{
55         let nextAmount, nextRecipient;
56
57         beforeEach(() =>{
58             nextAmount = 20;
59             nextRecipient = 'n3xt-4ddr355';
60             transaction = transaction.update(wallet, nextRecipient, nextAmount);
61         });
62
63         it('subtrancs the next amount from the senders output', ()=>{
64             expect(transaction.outputs.find(output => output.address === wallet.publicKey).amount)
65                 .toEqual(wallet.balance - amount - nextAmount);
66
67         });
68
69         it('outputs an amount for the next recipient', ()=>{
70             expect(transaction.outputs.find(output => output.address == nextRecipient).amount)
71                 .toEqual(nextAmount);
72         });
73     });
74 });
75
```