```
const SHA256 = require('crypto-js/sha256');
const EC = require('elliptic').ec;
var ec = new EC('secp256k1');

// Class for individual transactions including signatures
class Transaction{
    constructor(customerPubKey, merchantPubKey, amount){
        this.customerPubKey = customerPubKey;
        this.merchantPubKey = merchantPubKey;
        let date = new Date(Date.now());
        this.transDate = date.getMonth() + date.getDate() + date.getYear();
        this.amount = amount;
        this.customerSignature;
        this.merchantSignature;
    }

    // Apply the Customers Signature
    CustomerSign(customerPrivKey){
        // Concatenate all data together into a single string
        let fields = this.customerPubKey;
        fields += this.merchantPubKey;
        fields += this.transDate;
        fields += this.amount;

        // Get the one-way hash of this string
        let hash = SHA256(fields);
        // Encrypt the Hash with the Private Key
        this.customerSignature = customerPrivKey.sign(hash.words);
    }

    // Apply the Merchants Signature
    MerchantSign(merchantPrivKey){
        // Concatenate all data together into a single string
        let fields = this.customerPubKey;
        fields += this.merchantPubKey;
        fields += this.transDate;
        fields += this.amount;
        fields += this.customerSignature;

        // Get the one-way hash of this string
        let hash = SHA256(fields);
        // Encrypt the Hash with the Private Key
        this.merchantSignature = merchantPrivKey.sign(hash.words);
    }
}

class Miner{
    constructor(){
        this.privateKey = ec.genKeyPair();
        this.publicKey = this.privateKey.getPublic();
    }

    signTransaction(transaction){
        let customerSignature = transaction.customerSignature;
        let merchantSignature = transaction.merchantSignature;
        let content = { customerSignature, merchantSignature};
        let hash = SHA256(content);
        return this.privateKey.sign(hash.words);
    }
}

class Block{
    constructor(index, data, previousHash, minerSignature){
        this.index = index;
        this.data = data;
        this.previousHash = previousHash;
        this.minerSignature = minerSignature;
        this.hash = this.calcualteHash();
    }

    calcualteHash(){
        //return SHA256(this.index + this.data + this.previousHash + this.data.customerSignature + this.data.merchantSignature).toString();
        return SHA256(this.index + JSON.stringify(this.data), this.previousHash).toString();
    }


}

class BlockChain{
    constructor(){
        this.chain = [this.createGenesisBlock()];
    }
```

```
createGenesisBlock(){
    return new Block(0, "11/06/1988", "Genesis block", "0");
}

getLatestBlock(){
    return this.chain[this.chain.length-1]
}

addBlock(newBlock){
    newBlock.previousHash = this.getLatestBlock().hash;
    newBlock.hash = newBlock.calcualteHash();
    this.chain.push(newBlock);
}

isChainValid(){
    console.log("chain.length = ", this.chain.length);
    for(let i=1; i<this.chain.length; i++){
        const currentBlock = this.chain[i];
        const previousBlock = this.chain[i-1];
        if(currentBlock.hash != currentBlock.calcualteHash()){
            return false;
        }
        if(currentBlock.previousHash != previousBlock.calcualteHash()){
            return false;
        }
    }
    return true;
}
}

// Generate a Random integer between min and max
function RandInt(min, max){
    let range = max - min;
    return Math.round(Math.random()*range) - min;
}

// Create a Customer Class
class Customer{
    constructor(id){
        this.id = id;
        let key = ec.genKeyPair();
        this.privateKey = key;
        this.publicKey = key.getPublic();;
    }
}

// Create a Merchant Class
class Merchant{
    constructor(id){
        this.id = id;
        let key = ec.genKeyPair();
        this.privateKey = key;
        this.publicKey = key.getPublic();;
    }
}

// Create a list of 5 Customers
let customers = [];
for(let i=0; i<5; i++){
    let customer = new Customer(i);
    customers.push(customer);
}

// Create a list of 2 Merchants
let merchants = [];
for(let i=0; i<2; i++){
    let merchant = new Merchant(i);
    merchants.push(merchant);
}

// Instantiate a Coin Miner
var pickAxe = new Miner();

// Implement the Block Chain
let trippCoin = new BlockChain();

// Generate 25 Random Transactions
let transactions = [];
for(let i=0; i<25; i++){
    let amount = RandInt(0, 300);
    let merchant = merchants[RandInt(0, merchants.length - 1)];
    let customer = customers[RandInt(0, customers.length - 1)];
```

```
    let transaction = new Transaction(customer.publicKey, merchant.publicKey, amount);

    // Apply Customer and Merchant Signatures
    transaction.CustomerSign(customer.privateKey);
    transaction.MerchantSign(merchant.privateKey);

    transactions.push(transaction);
    let previousHash = trippCoin.chain[i];
    let minerSignature = pickAxe.signTransaction(transaction);
    let block = new Block(i + 1, transaction, previousHash, minerSignature);
    trippCoin.addBlock(block);
}

console.log("\n\n");
// (1) Print Transactions 1-4
console.log("(1)");
for(i=0; i<4; i++){
    let transaction = transactions[i];
    console.log("Merchant Public Key: ", JSON.stringify(transaction.merchantPubKey));
    console.log("Customer Public Key: ",  JSON.stringify(transaction.customerPubKey));
    console.log("Transaction Date: ",  transaction.transDate);
    console.log("Amount: ",  transaction.amount);
    console.log("\n");
}

// (2) Increment Ammount of Tranaction #15 by
console.log("(2)")
let blockIdx = 15;
console.log("Original Amount =  ", trippCoin.chain[blockIdx].data.amount);
// Check the validity of the BlockChain
//console.log(trippCoin.chain[blockIdx], "\n\n");
console.log("Block Chain Validity = ", trippCoin.isChainValid());
console.log("\n\n");

// Tamper with the data
trippCoin.chain[2].data.amount += 10;
console.log("Tampered Amount = ", trippCoin.chain[blockIdx].data.amount);
// Check the validity of the BlockChain
//console.log(trippCoin.chain[blockIdx], "\n\n");
console.log("Block Chain Validity = ", trippCoin.isChainValid());

// (3) Search Through the Blockchain and print all transactions for Customer #3
console.log("(3)");
console.log("\n\n Customer # 3 Transactions: ");
customer3PubSig = customers[2].publicKey;
trippCoin.chain.forEach((block) => {
    if(block.data.customerPubKey == customer3PubSig){
        let transaction = block.data;
        console.log("Merchant Public Key: ", JSON.stringify(transaction.merchantPubKey));
        console.log("Customer Public Key: ",  JSON.stringify(transaction.customerPubKey));
        console.log("Transaction Date: ",  transaction.transDate);
        console.log("Amount: ",  transaction.amount);
        console.log("\n");
    }
});

// (4) Search Through the Blockchain and print all transactions for Merchant #2
console.log("(4)");
console.log("\n\n Merchant # 2 Transactions: ");
merchant2PubSig = merchants[1].publicKey;
trippCoin.chain.forEach((block) => {
    if(block.data.merchantPubKey == merchant2PubSig){
        let transaction = block.data;
        console.log("Merchant Public Key: ", JSON.stringify(transaction.merchantPubKey));
        console.log("Customer Public Key: ",  JSON.stringify(transaction.customerPubKey));
        console.log("Transaction Date: ",  transaction.transDate);
        console.log("Amount: ",  transaction.amount);
        console.log("\n");
    }
});
```