Name	Kareem mohamed mohamed
Section	3
Code	220100399

Lab 3

```
"use strict";


// required npm install blind-signatures
const blindSignatures = require('blind-signatures');
const { Coin, COIN_RIS_LENGTH, IDENT_STR, BANK_STR } = require('./coin.js');
const utils = require('./utils.js');


// Generate bank key
const BANK_KEY = blindSignatures.keyGeneration({ b: 2048 });
const N = BANK_KEY.keyPair.n.toString();
const E = BANK_KEY.keyPair.e.toString();


/**
 * Sign a coin using the bank's private key.
 */
function signCoin(blindedCoinHash) {
  return blindSignatures.sign({
    blinded: blindedCoinHash,
    key: BANK_KEY,
  });
```

```
}

/**
 * Parse coin string into left/right hash arrays.
 */
function parseCoin(s) {
  let [cnst, amt, guid, leftHashes, rightHashes] = s.split('-');
  if (cnst !== BANK_STR) {
    throw new Error(` Invalid identity string: ${cnst} received, but ${BANK_STR} expected `);
  }
  return [leftHashes.split(','), rightHashes.split(',')];
}

/**
 * Simulate merchant accepting a coin and selecting one side of the RIS.
 */
function acceptCoin(coin) {
  // Verify signature
  const isValid = blindSignatures.verify({
    unblinded: coin.signature,
    key: BANK_KEY,
    message: coin.toString(),
  });

  if (!isValid) {
    return new Error("Signature verification failed!");
```

```
  }

  const [leftHashes, rightHashes] = parseCoin(coin.toString());
  const selectLeft = Math.random() > 0.5;
  const selectedSide = selectLeft ? leftHashes : rightHashes;

  console.log(`\n[Merchant] Accepting coin ${coin.guid}`);
  console.log(`[Merchant] Selected side: ${selectLeft ? 'Left' : 'Right'}`);
  console.log(`[Merchant] RIS sample: ${selectedSide.slice(0, 3).join(', ')}...`);

  return selectedSide;
}

/**
 * Determine who cheated (merchant or purchaser).
 */
function determineCheater(guid, ris1, ris2) {
  console.log(`\n[Bank] Checking double-spend for coin: ${guid}...`);

  if (JSON.stringify(ris1) === JSON.stringify(ris2)) {
    console.log("[Bank] No double spending detected or RIS values are identical.");
    return;
  }

  for (let i = 0; i < ris1.length; i++) {
    const xorResult = utils.hash(utils.decryptOTP({
```

```
    key: Buffer.from(ris1[i], 'hex'),
    ciphertext: Buffer.from(ris2[i], 'hex'),
    returnType: 'string'
   }));

   if (xorResult.startsWith(utils.hash(IDENT_STR))) {
    console.log(`[Bank] Purchaser is the cheater! Identity: ${xorResult}`);
    return;
   }
  }

  console.log("[Bank] RIS mismatch doesn't reveal identity. Merchant may be the cheater.");
}



// --------------------------------
// Main flow
// --------------------------------

let coin = new Coin('alice', 20, N, E);
coin.signature = signCoin(coin.blinded);
coin.unblind();

let ris1 = acceptCoin(coin);
let ris2 = acceptCoin(coin);
```

```
// Case 1: Same coin used twice
determineCheater(coin.guid, ris1, ris2);

// Case 2: Compare same RIS (no cheating)
determineCheater(coin.guid, ris1, ris1);
```