Setting up an Anonymous Blockchain

Name: Chia Tze Hank

Course: Meng Electrical & Electronics Engineering

Supervisor: Dr Khor Jing Huey

University of Southampton Malaysia Campus

Author Note

This report is submitted upon completion of Summer Research Internship at University of Southampton Malaysia Campus

Abstract

A blockchain is a record of transactions published to the network maintained through a consensus algorithm of multiple computers in a peer-to-peer network. However, most of the transactions do not have privacy as details of the transactions such as the sender, receiver and amount are needed to be included in the hashing algorithm for security purposes. An anonymous blockchain maintains the security aspect of a general blockchain, while utilizing other cryptography techniques such as ring signatures, pedersen commitments and dual-key stealth address protocol to secure the privacy of the users on the blockchain. These cryptography technologies usually use elliptic curve properties to verify the integrity of a content without revealing the details of the transaction.

Table of content

1. Introduction
   1. Aim
   2. Objectives
2. Setting up an Anonymous Blockchain System
   1. Basic Blockchain Setup
      1. Setting up a node
      2. Setting up user interface
      3. Blockchain interactions
   2. Anonymizing Sender’s Address
      1. Ring Signature
   3. Anonymizing Transaction Amount
      1. Pedersen Commitment
      2. Range Finding and Balance Checking
   4. Anonymizing Receiver’s Address
      1. Dual-Key Stealth Address Protocol (DKSAP)
3. Conclusion

1. Introduction

This document is a report of creating an anonymous blockchain using the fundamentals of a public blockchain. The project attempts to anonymize the sender, receiver and amount of value send in transactions using ring signatures, pedersen commitments and stealth addresses.

1.1 Aim

Our aim is to create a full self-operating anonymous blockchain while maintaining the security of the blockchain.

1.2 Objectives

1. Create the fundamentals of a working public blockchain.
2. Add Ring Signature to anonymize the sender’s identity.
3. Add Pedersen Commitment to anonymize amount of transaction.
4. Add Dual-Key Stealth Address to anonymize the sender and receiver’s identity.

2. Setting up an Anonymous Blockchain System

2.1 Basic Blockchain Setup

2.1.1 Setting up a node

First, download the project <https://github.com/HankRobot/LoRaBlockchain>. After downloading the projects, we need to install NodeJS and typescript so we can run the code as the majority of the project uses JavaScript and Typescript libraries. Download Nodejs here -> <https://nodejs.org/en/> and then in the command line type:

$ npm install typescript -g

A reference for typescript tutorials can be found [here](https://www.tutorialspoint.com/typescript/index.htm). After installing NodeJS and typescript, open the command line in the project directory naivecoin-master and type the following:

$ npm install

$ npm start

The project will begin installing all the required libraries and start compiling the project after that. You should have your project hosted on <http://localhost:3001/blocks> , where all the transaction information are shown in JSON format, you can visualize them easier by downloading the chrome extension [JSON Formatter](https://chrome.google.com/webstore/detail/json-formatter/bcjindcccaagfpapjjmafapmmgkkhgoa).

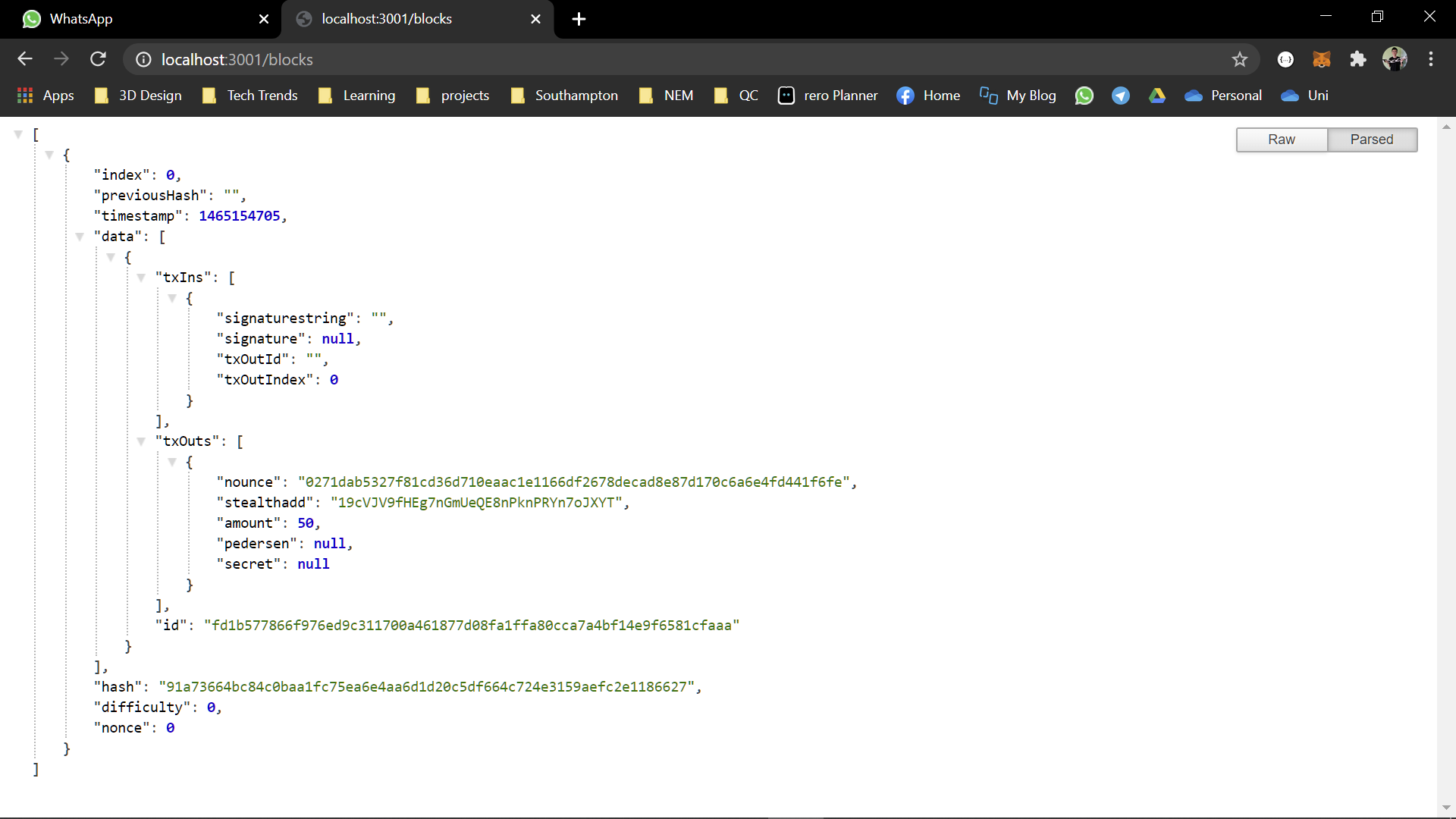


Figure : An anonymous blockchain with JSON Formatter

2.1 Basic Blockchain Setup

2.1.2 Setting up user interface

Next, setup the user interface of the blockchain by opening another command line in the project directory naivecoin-ui-master/naivecoin-wallet and type in the following:

$ npm install

$ npm start

This time, the project will be running on <http://localhost:8080/#/>, make sure naive-coin master is running whenever you run this project, otherwise you wont be able to interact with the blockchain. Both projects only need npm install once, and to start the project always use npm start.

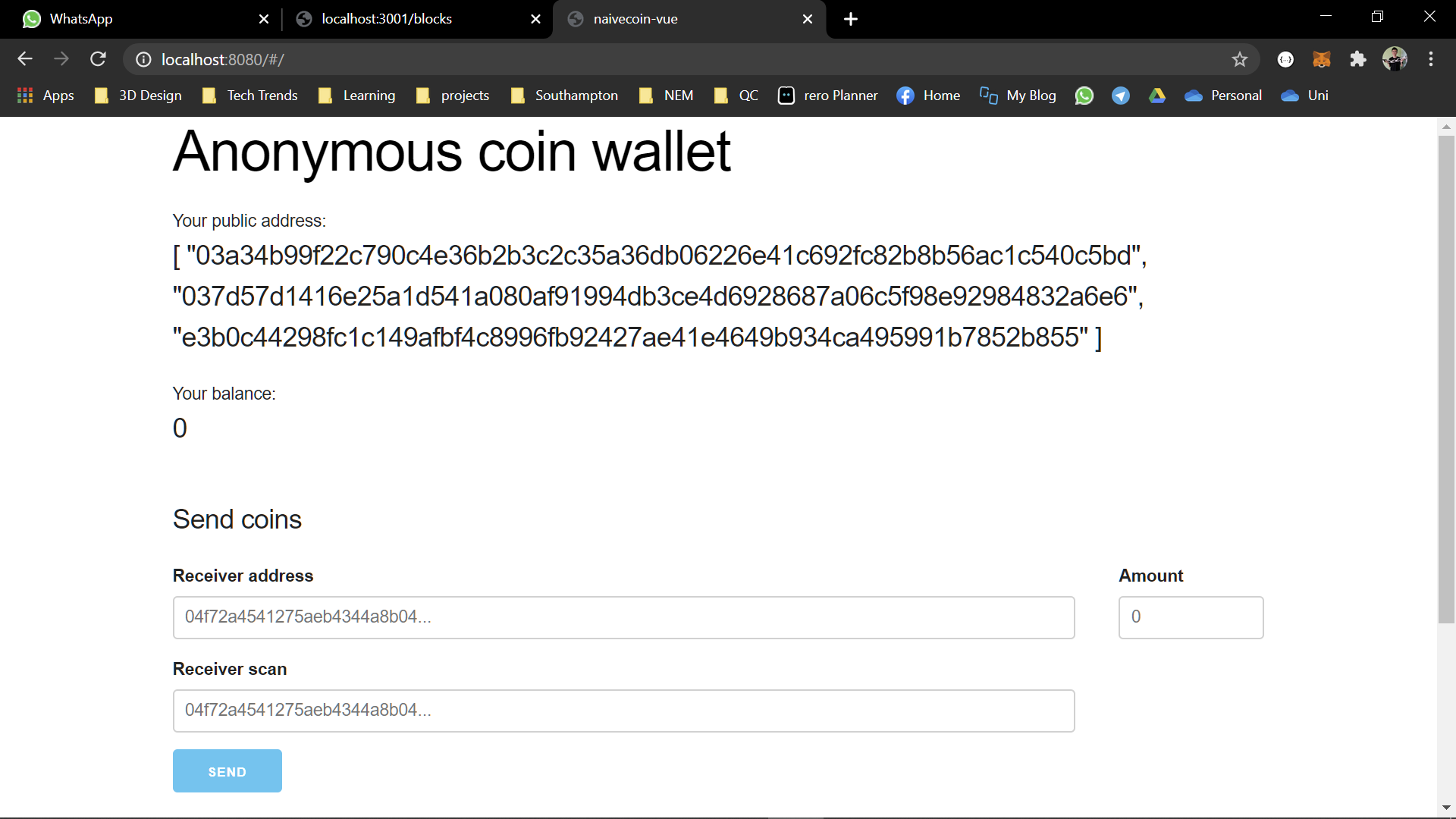


Figure 2: naivecoin-master basic user interface

A user-interface for visualizing the blockchain (without using JSON formatter) is also available in the project directory naivecoin-ui-master\naivecoin-explorer. The method to setup is also the same as the previous projects and is hosted on <http://localhost:8081/#/>.

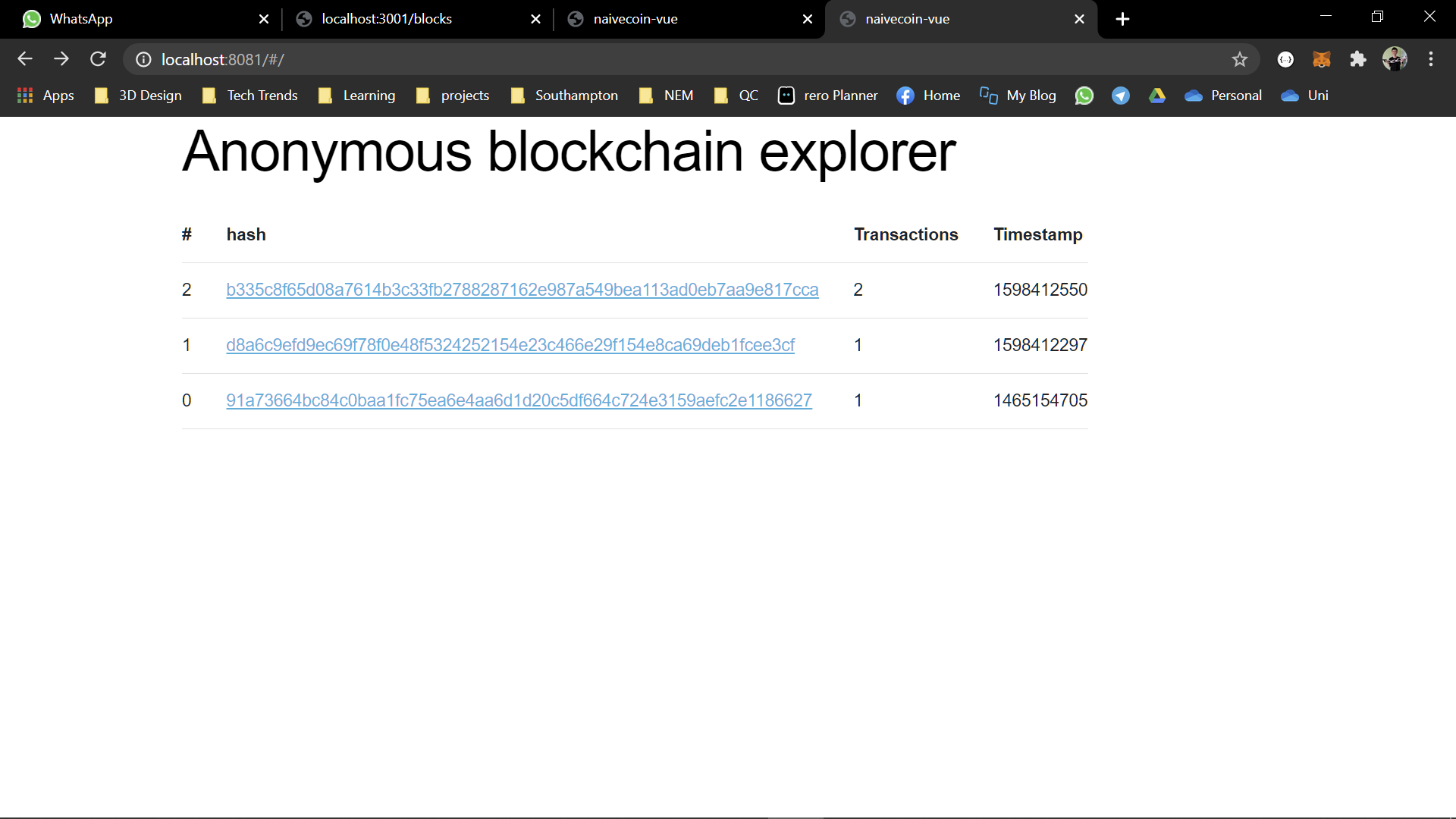


Figure 3: Main page of Anonymous blockchain explorer

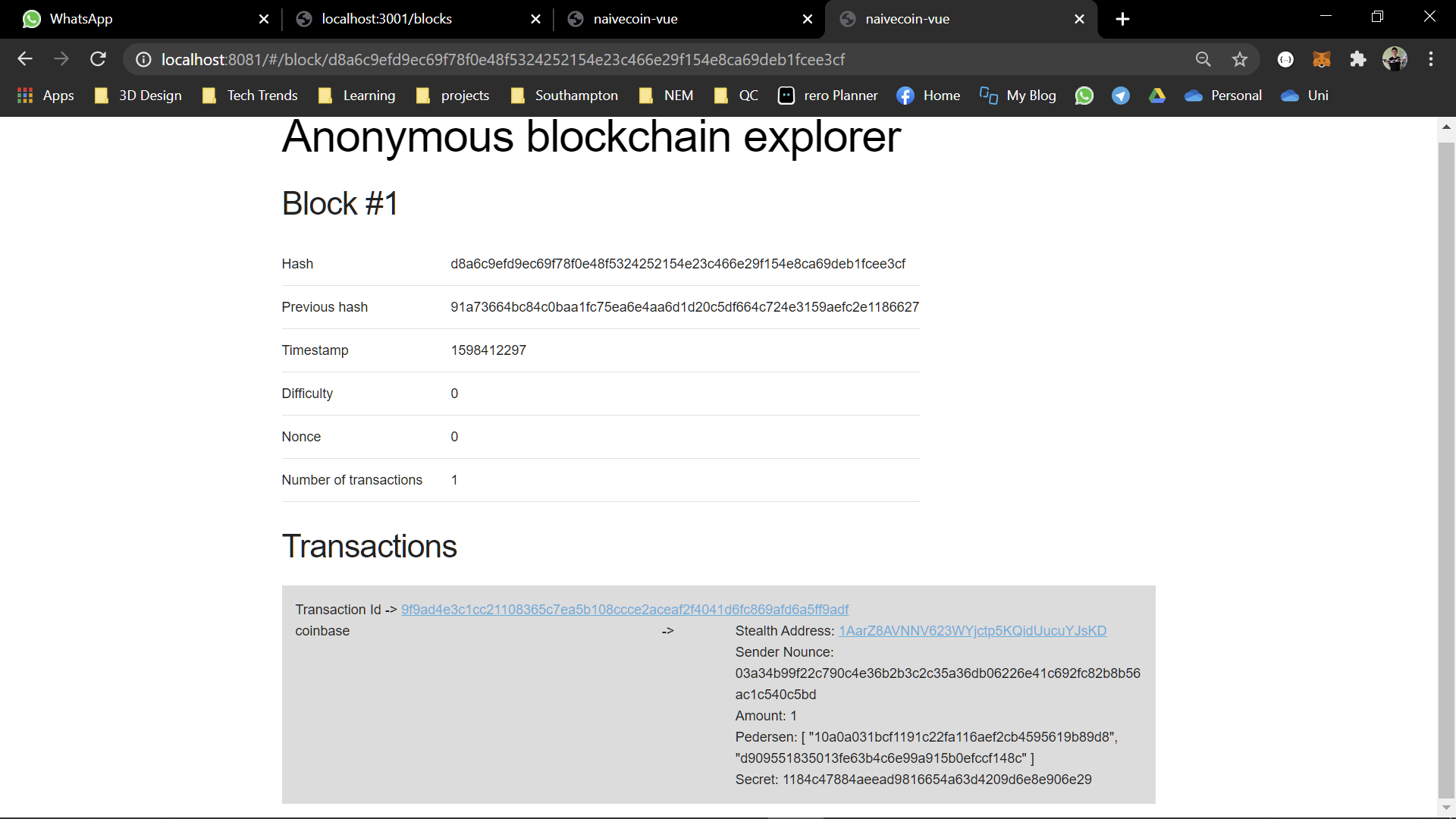


Figure 4: Details of a block in the explorer

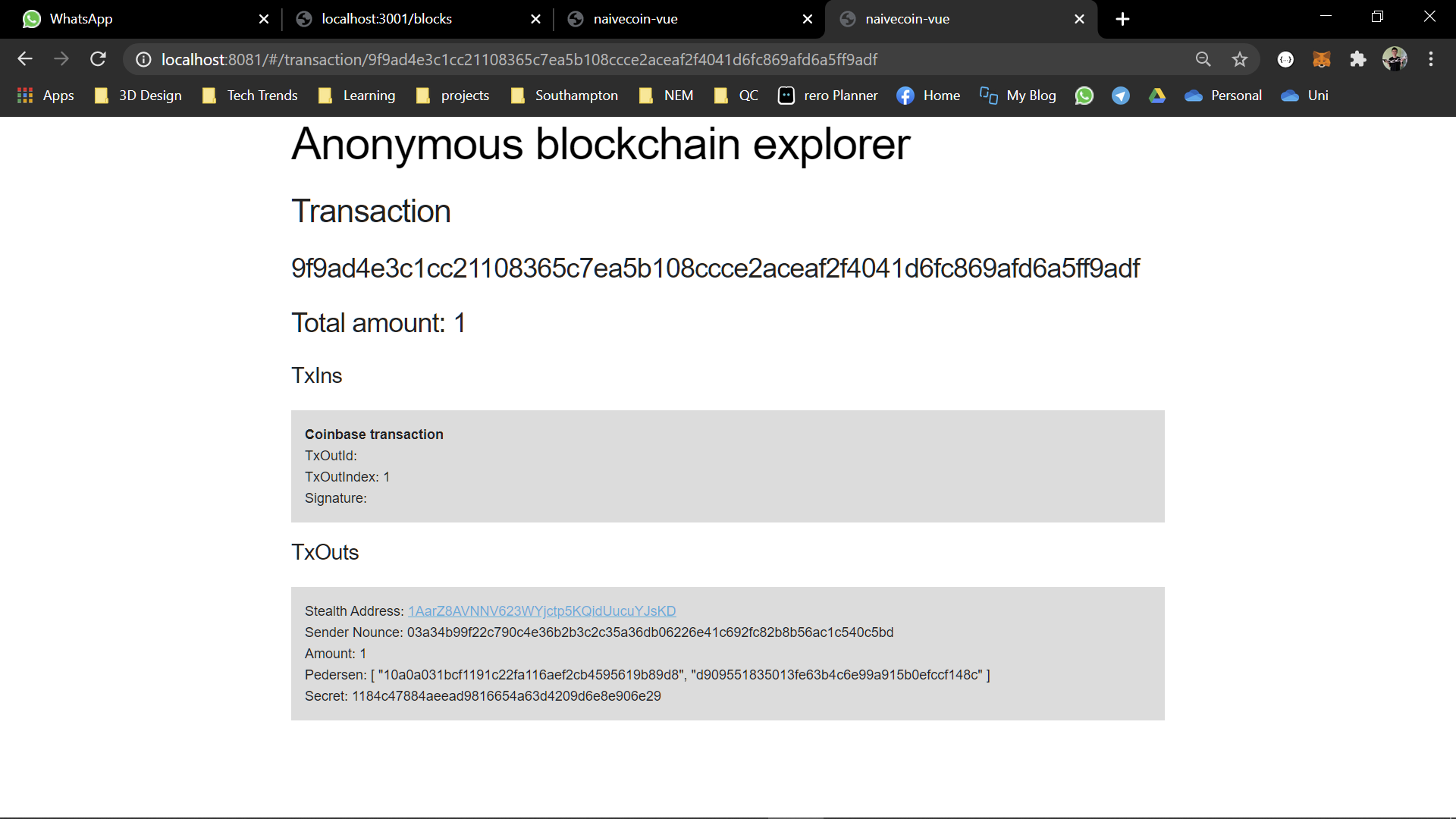


Figure 5: Details of txin and txout based on transaction id

2.1 Basic Blockchain Setup

2.1.3 Blockchain Interactions

Before we start a transaction, our account needs to have a balance to send money to other accounts. Initially, your wallet should have a balance of 0, click the button “click to mine block” to add coins to your wallet. You should have 50 coins in your wallet after you mined your first block, this transaction is known as coinbase transaction. Click the refresh button to see the new block mined on <http://localhost:3001/blocks>:

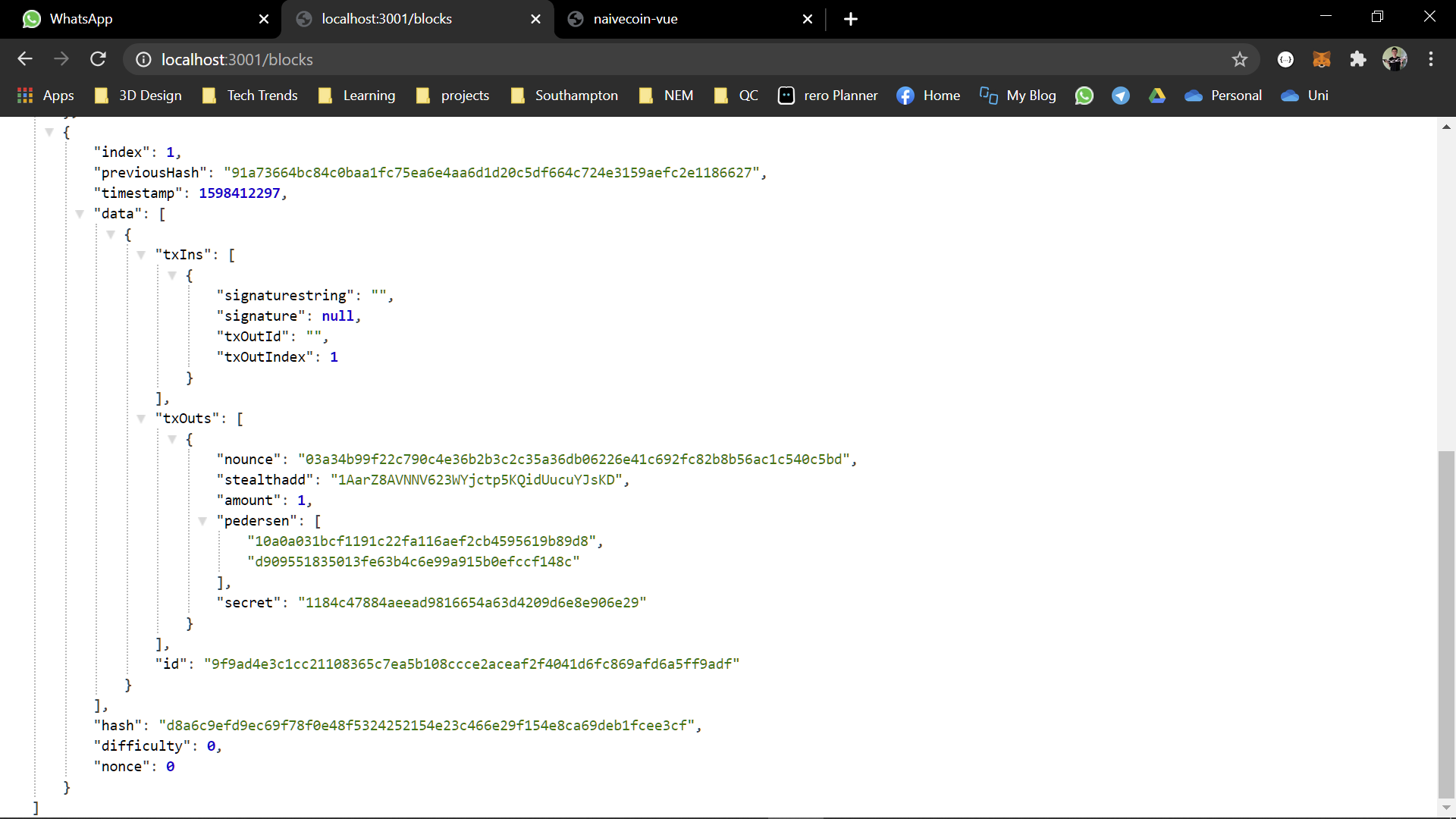


Figure 6: Your first coinbase transaction

The first block of a blockchain is known as a genesis block, and usually it involves a coinbase transaction where the miner is rewarded for creating the first block.   
key: 0271dab5327f81cd36d710eaac1e1166df2678decad8e87d170c6a6e4fd441f6f  
scan: 02f117f201530e78fc7e997694f69c6a15c3866743c4ec385fb07fc713946059b9

Copy and paste the public scan and key into the user interface as follow:

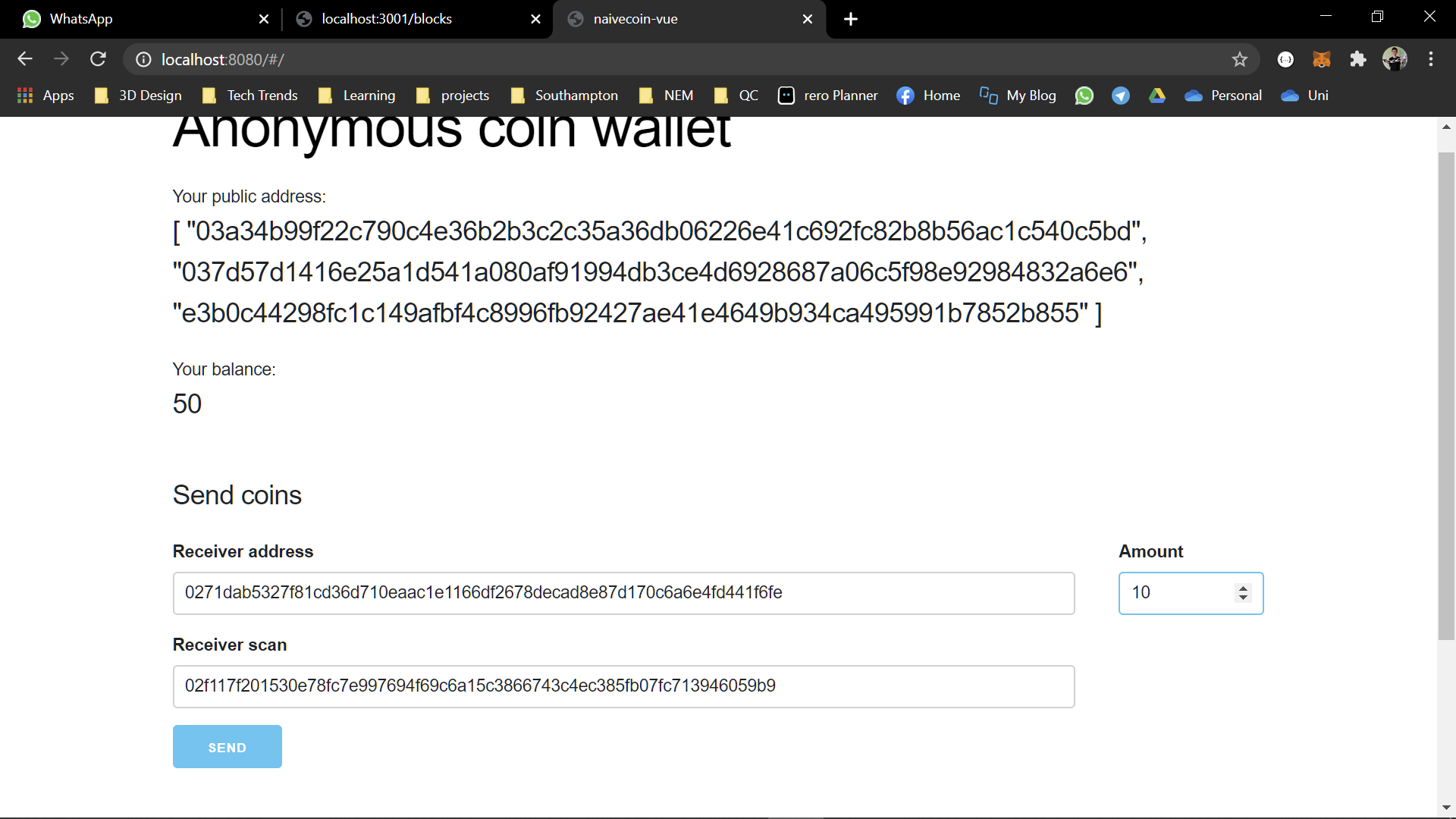


Figure 7: Sending Transactions to another account

Click the send button, you will see the transaction pool showing your transaction, this is known as an unconfirmed transaction and has to be mined to be announced to the blockchain. Click the mine block button and refresh the <http://localhost:3001/blocks> page again.

Based on Figure 5, there are two sets of transactions in the block, one is a coinbase transaction (where the txin does not have a signature and txoutid) and a user transaction between the sender and receiver. Notice that both the sender and receiver address are anonymous on the public blockchain, as well as the exact amount of transaction available by the sender and amount the receiver receives.

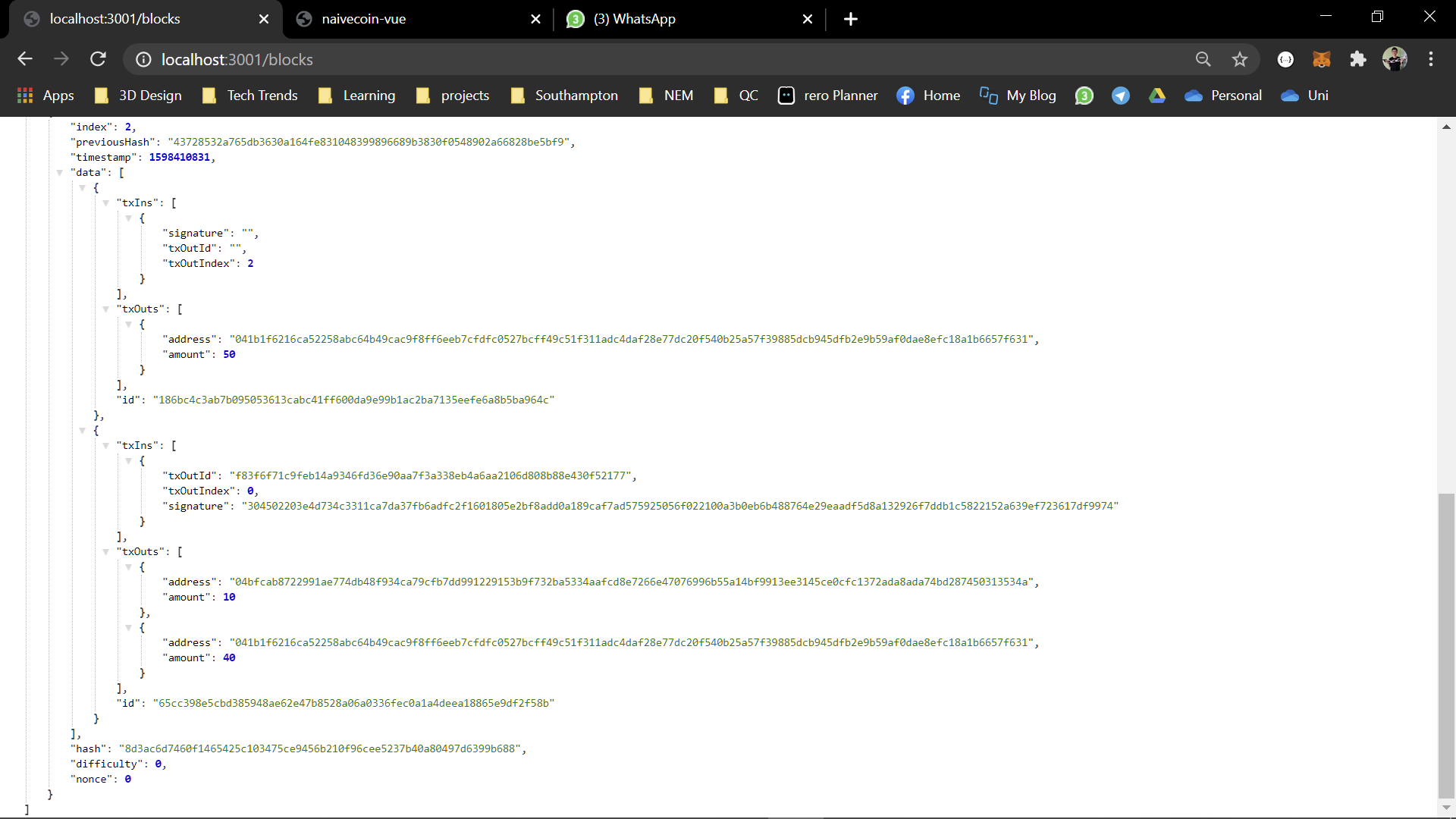


Figure 8: A successful blockchain transaction

Notice that your wallet now has 90 coins (due to 10 coins transferred to another user and mining a block rewards you 50 coins) but none of this information are obviously shown on the blockchain.

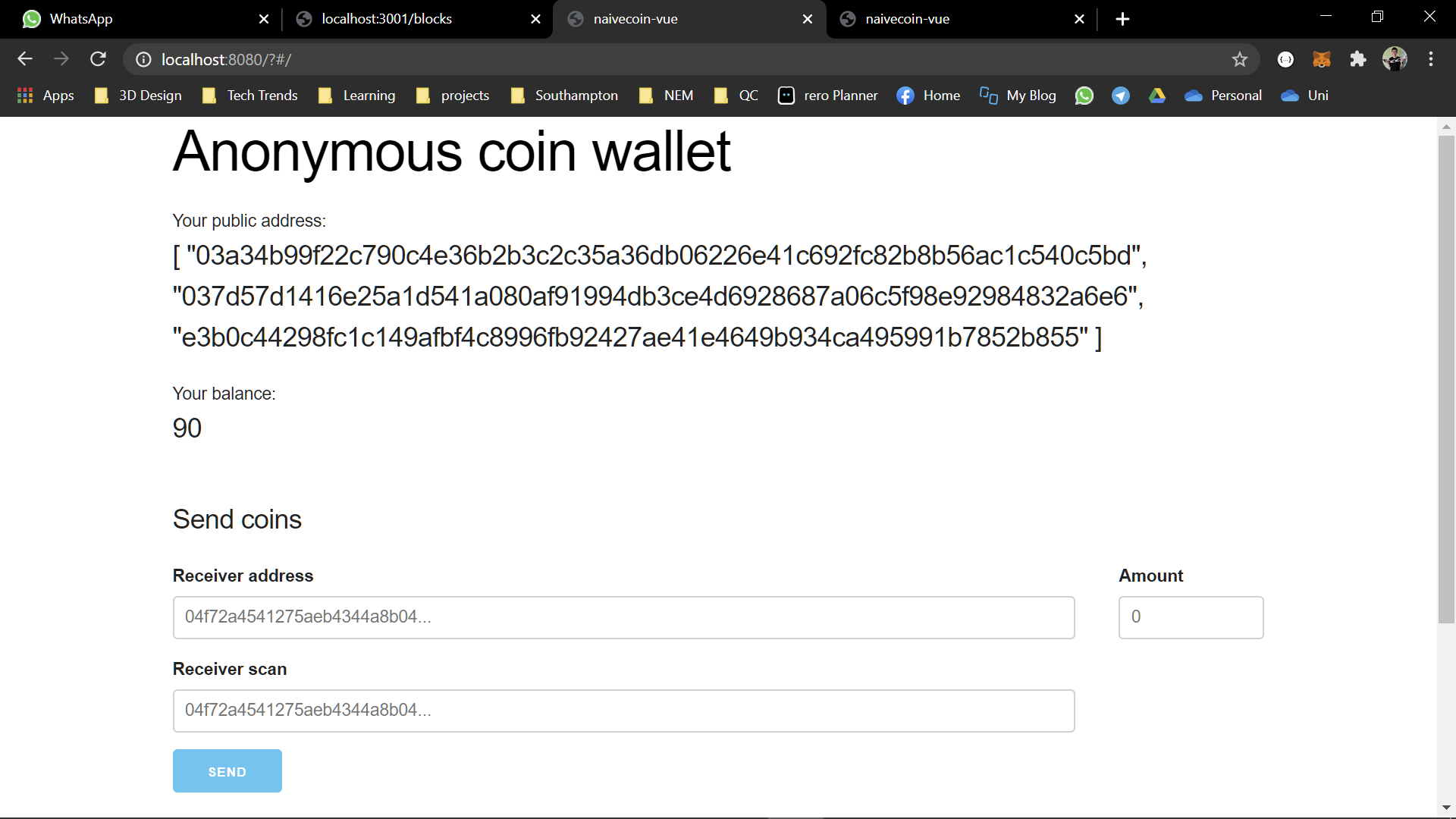


Figure 9: Resulting balance after transaction

2.2 Anonymizing Sender’s Address

2.2.1 Ring Signature

A ring signature is a type of digital signature that can be performed by any member of a group of users that each have keys. Hence, instead of a user representing a transaction address, a group is used to sign transactions. A good read about ring signatures can be found [here](https://web.getmonero.org/resources/moneropedia/ringsignatures.html). For the implementation of ring signatures, the source code can be found [here](https://github.com/mtsalenc/ot-ring-signature).



Figure 10: Example of Ring Signature Implementation

According to Figure 10, 4 users were used to develop a ring signature, and the signature and message can then be verified using the ring signature’s public key.

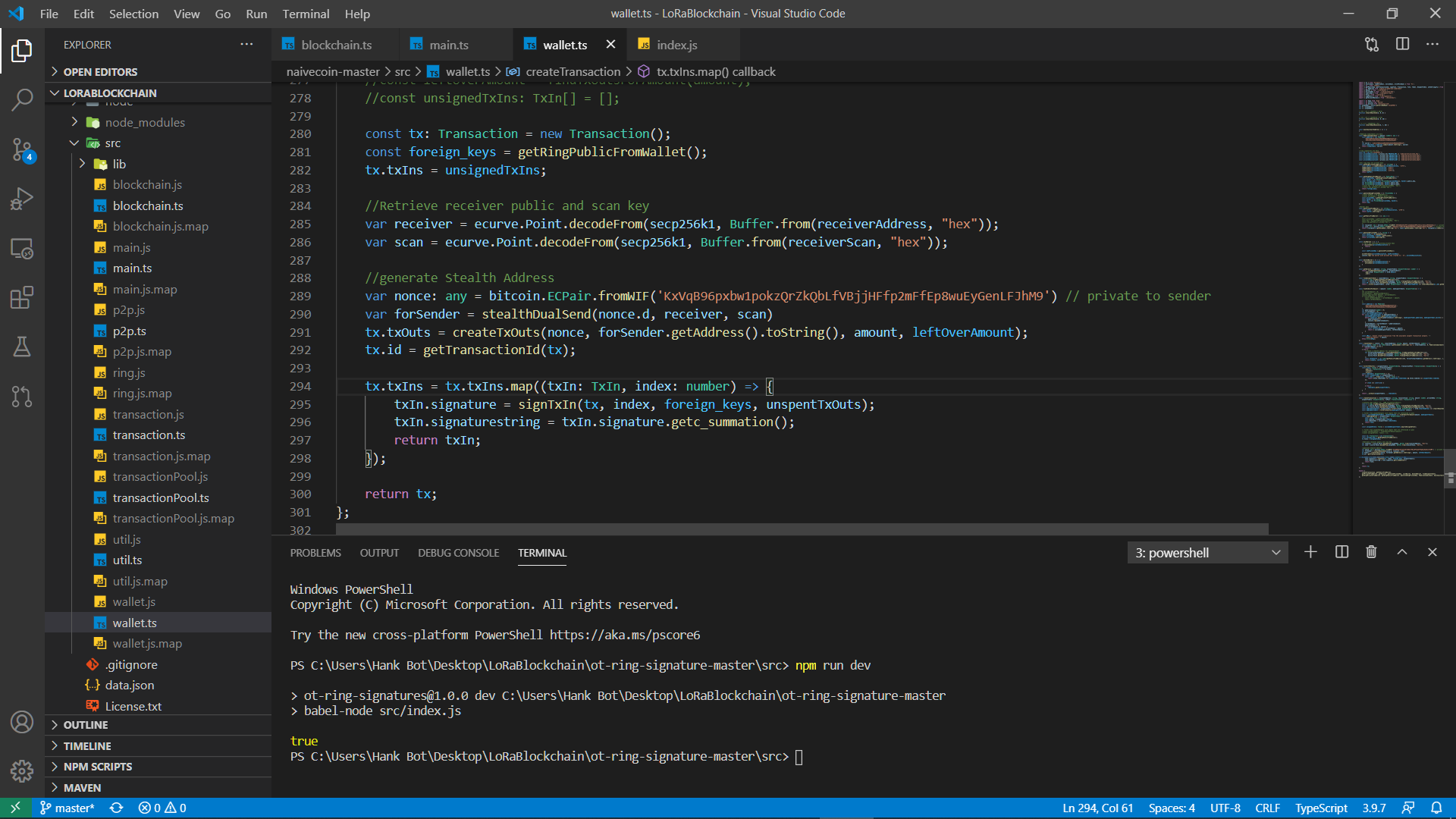


Figure 11: Ring Signature for TxIn Signature

In our anonymous blockchain, the ring signature is implemented in wallet.ts, functions createTransaction(), here the txin signature is signed with the transaction id as the message, hence the source of the transaction is anonymous. However, due to the ring signature containing elliptic properties, an error pops up to display the entire signature, hence the c\_summation is only stored in the blockchain as verification as show as below:

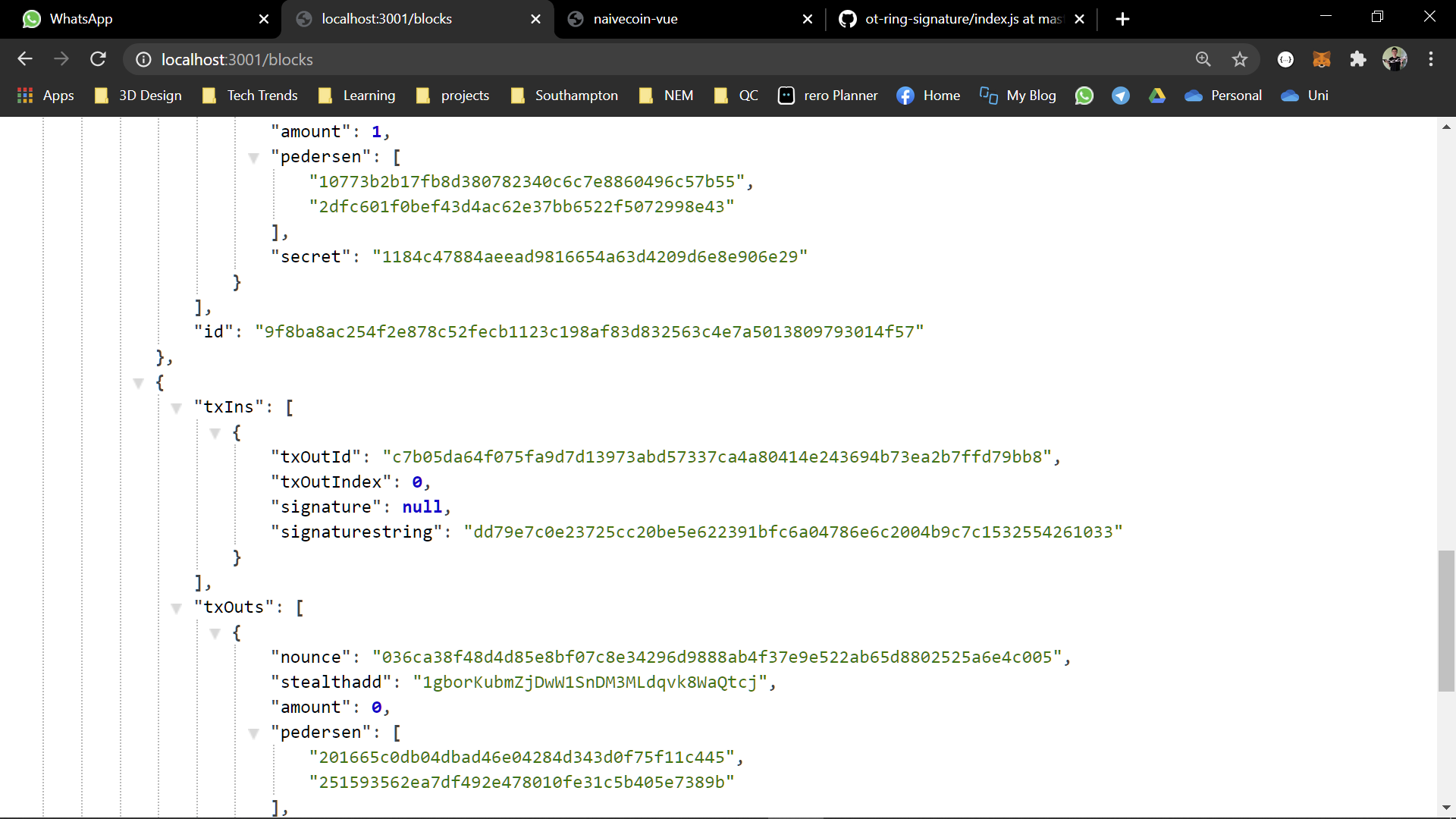


Figure 12: Ring Signature stored on the blockchain

Open up the file naivecoin-master/src/lib/signature.ts, function verify(), here we can see the message is combined with the public keys (this process is known as signing) and the result is compared with the c\_summation (verification). However, it would be best to involve the entire ring signature in the future if possible.

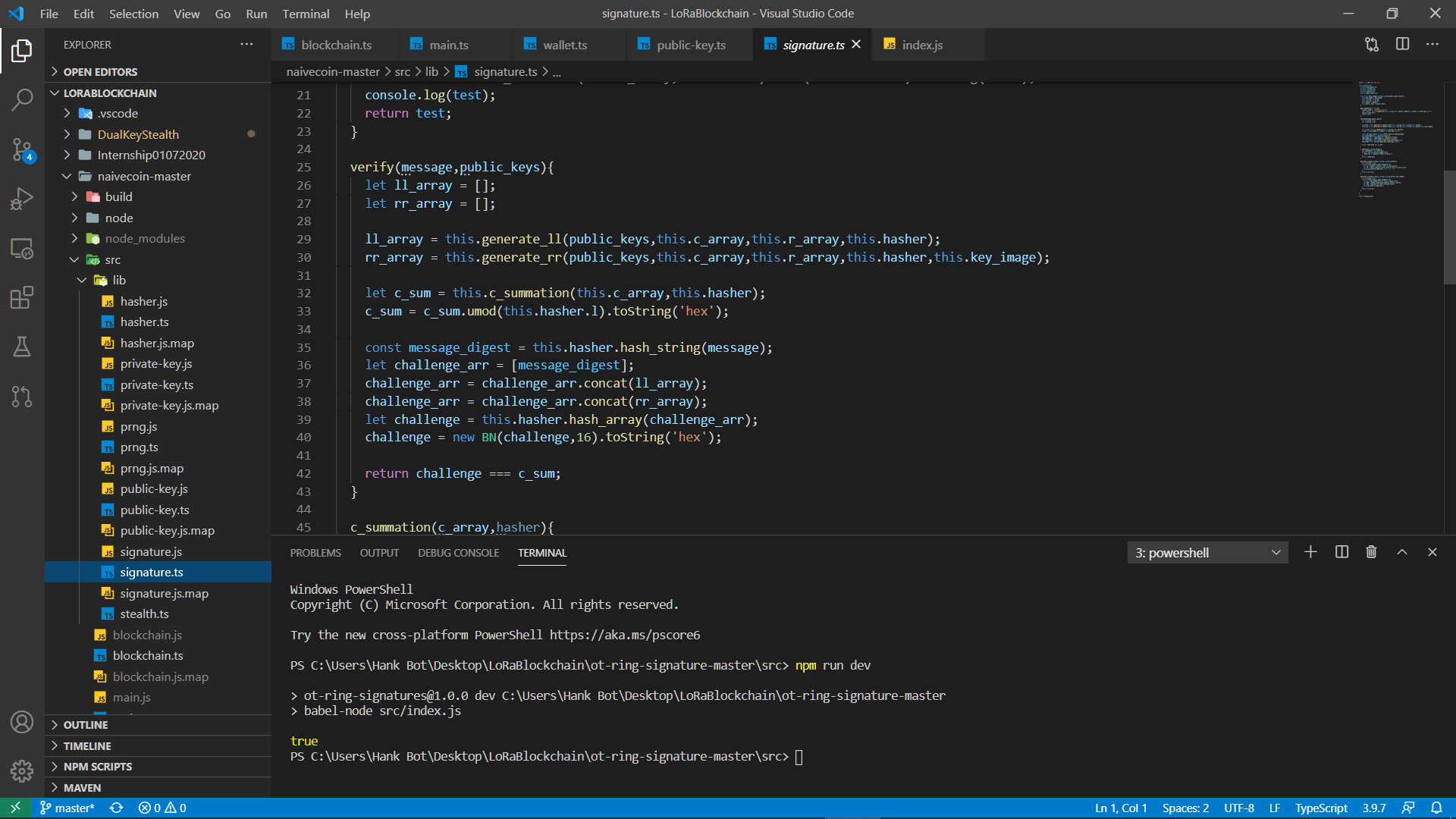


Figure 13: Verification of Ring Signature

Open up the file naivecoin-master/src/transaction.ts, function validateTxIn(). Based on Figure 14 below, the transaction id is received, and the public keys are used to sign the transaction id again, if the verification is correct, the transaction is officially announced to the chain. A way to improve this is to retrieve the c\_summation from the block and attempt to parse into the generated ring signature, however this would require an edit of the ring signature library.

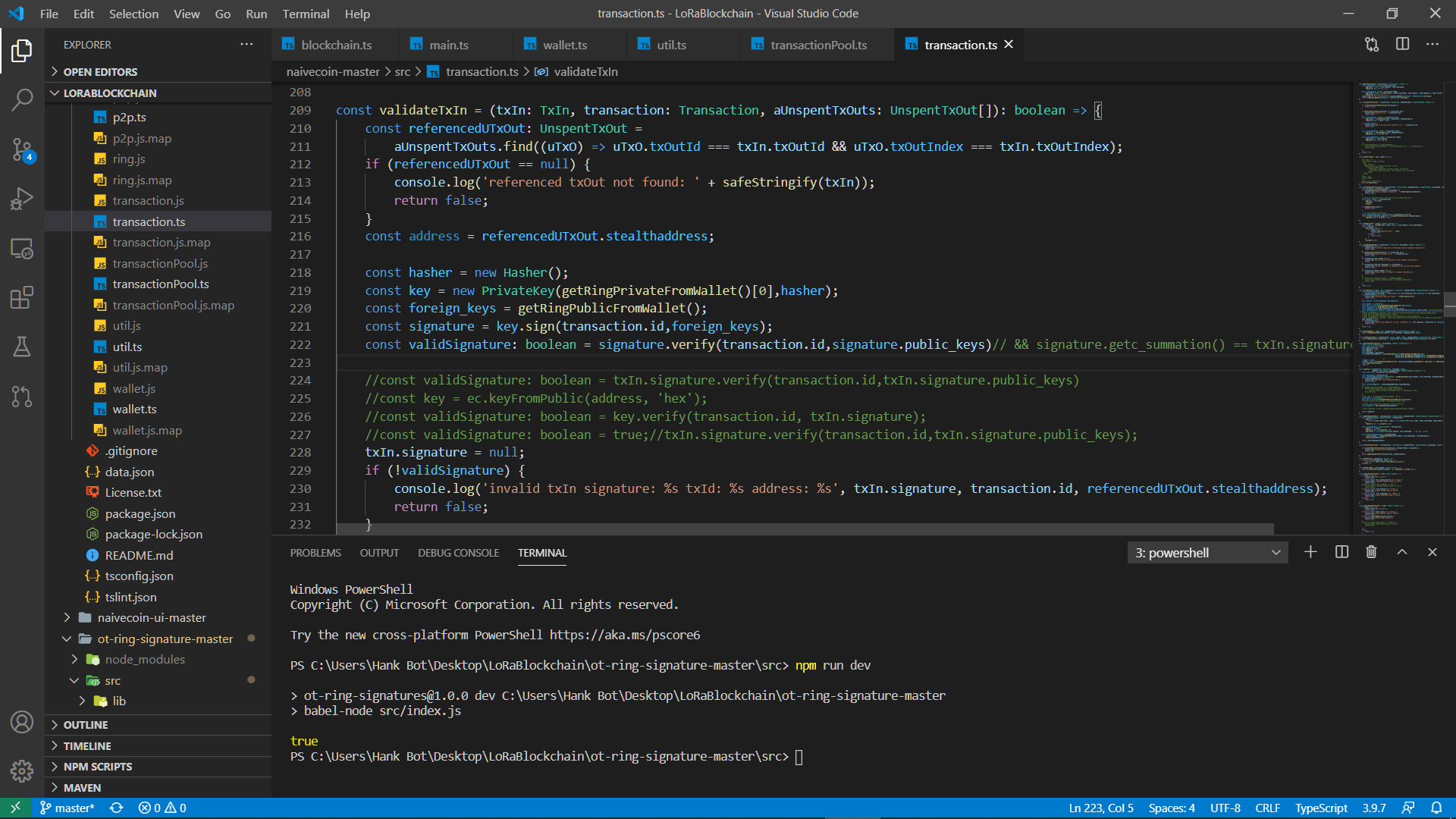


Figure 14: Verification of Ring Signature in Blockchain

2.3 Anonymizing Transaction Amount

2.3.1 Pedersen Commitment

A Pedersen commitment is a commitment that has the property of being additive. If C(a) and C(b) denote the commitments for values a and b respectively, then C(a + b) = C(a) + C(b). This property is useful when committing transaction amounts, as one could prove, for instance, that inputs equal outputs, without revealing the amounts at hand. More information can be found at <https://pdfs.semanticscholar.org/16a8/c0aa45bd09830b8c5115c2c1e441f177fc82.pdf> page 21.

The pedersen commitment source code is from [here](https://www.npmjs.com/package/simple-js-pedersen-commitment). The implementation of pedersen commitment is simple, according to the figure below, a pedersen is generated with each transaction amount and then announced to the blockchain (stored in txout) with the commitment and secret for verification in range finding and balance checking. The code can be found in wallet.ts.

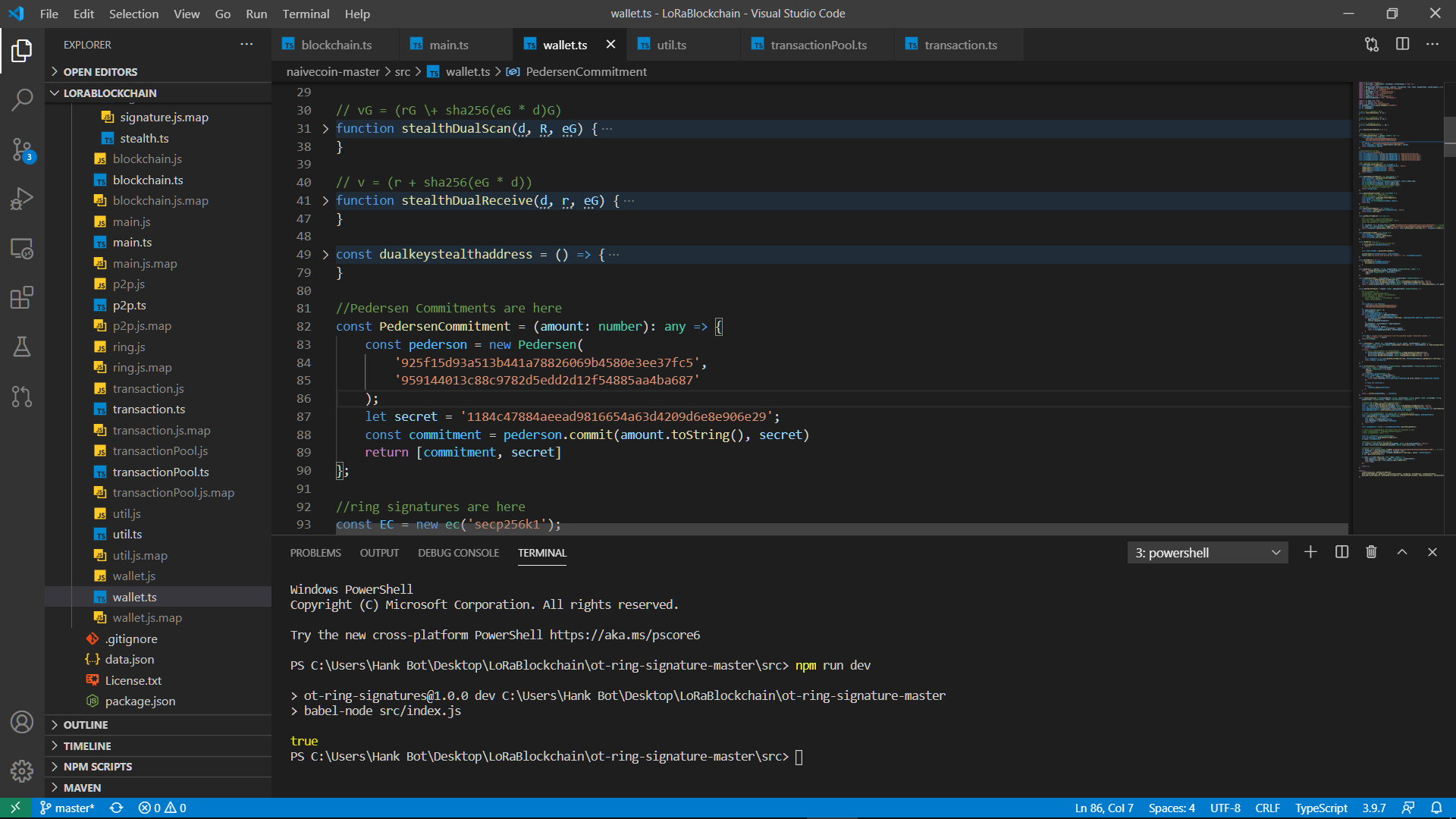


Figure : Pedersen Commitment on Blockchain

Based on Figure 16, the pedersen commitment and secret are stored in the txOuts, with amount indicating whether the transaction is to the sender/receiver (0 is receiver, 1 is sender). The amounts published on the chain are therefore anonymous and can verified for the correct amount.

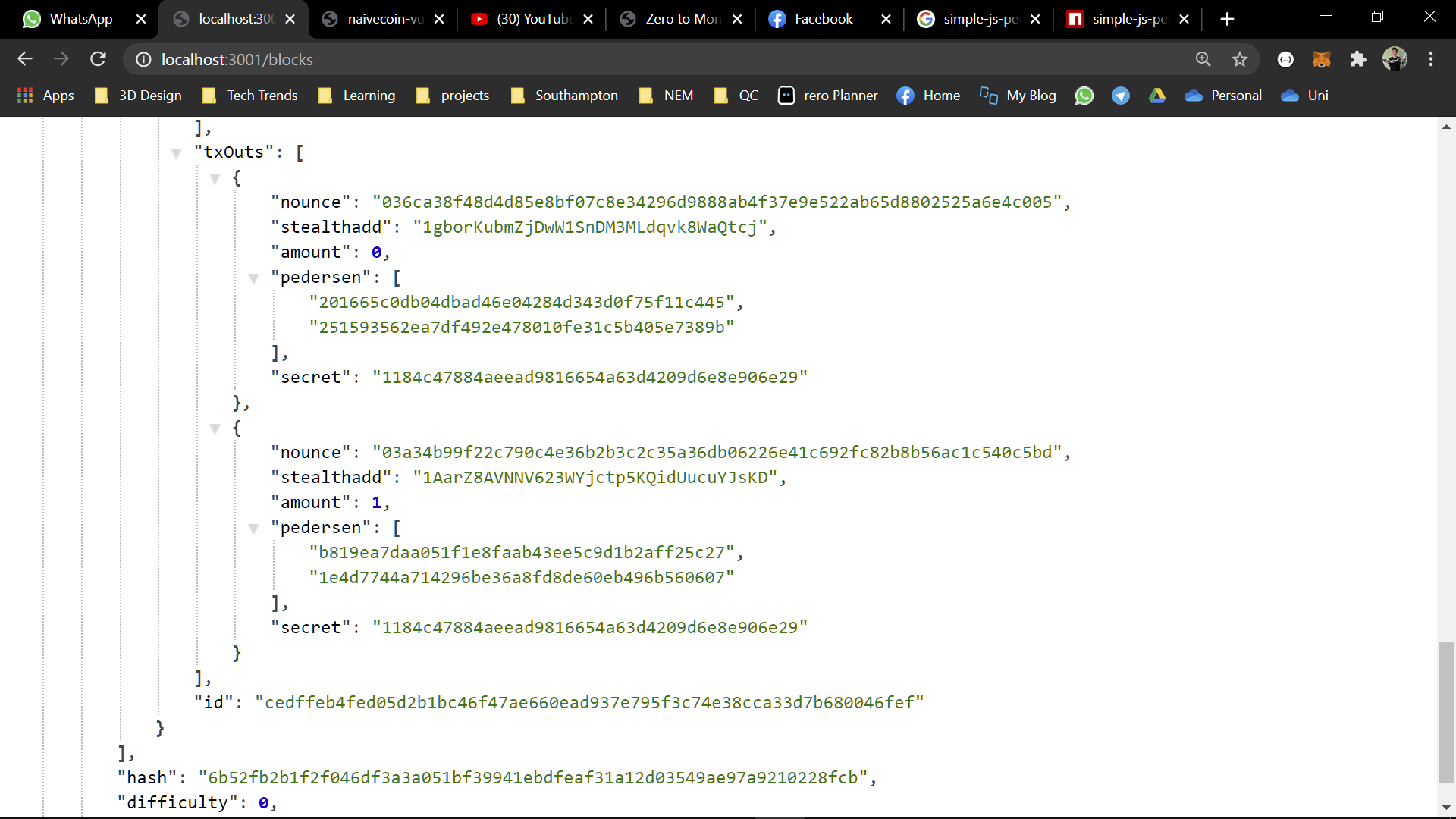


Figure : Pedersen Commitment shown on block

2.3 Anonymizing Transaction Amount

2.3.2 Range Finding and Balance Checking

To find the total balance available, the stealth address of the user is checked (more on this later) and all unspenttxout is filtered into a list, each unspenttxout has a pedersen and secret, if the amount, pedersen and secret returns true during verification, the transaction amount is validated.

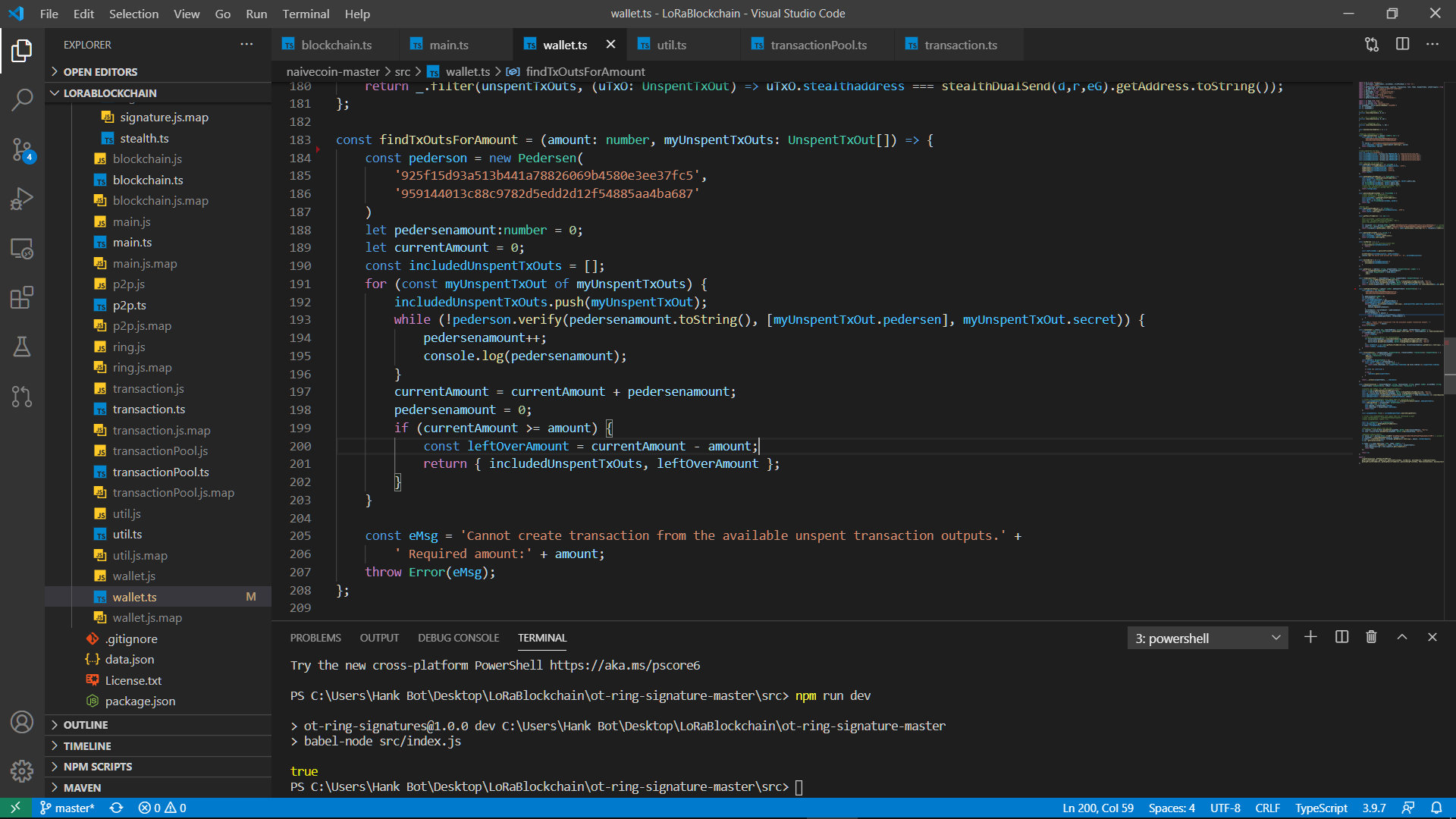


Figure : Range Finding and Balance Checking

2.4 Anonymizing Receiver’s Address

2.4.1 Dual-Key Stealth Address Protocol (DKSAP)

The receiver has two keys known as a public and scan key, the keys are generated using an elliptic curve group. The stealth address generated is a combination of the sender’s public key, and the receiver’s public scan key in a variety of combinations below:  
1) Sender announces a stealth address using the receiver’s public and scan’s Q value, and his own d value from his key.  
2) Scanner announces a stealth address using the receiver’s public Q, his scan’s d value and the sender’s Q value.  
3) The receiver announces a stealth address using the sender’s Q value, and his own public and scan key’s d value. Only this stealth address can have a private key generated to sign transactions.  
  
All three stealth address must be the same, but only the receiver’s stealth address generated can retrieve a private key to sign transactions. Hence, the sender first receives the receiver’s public and scan key’s Q and uses his own d value. He announces the stealth address and his Q value to the blockchain, the receiver can then verify if the stealth address is correct (since he has his own d values and the sender’s Q value is announced to the blockchain), retrieve the private key and sign transactions with it. The anonymity of both the sender and receiver are then secured.

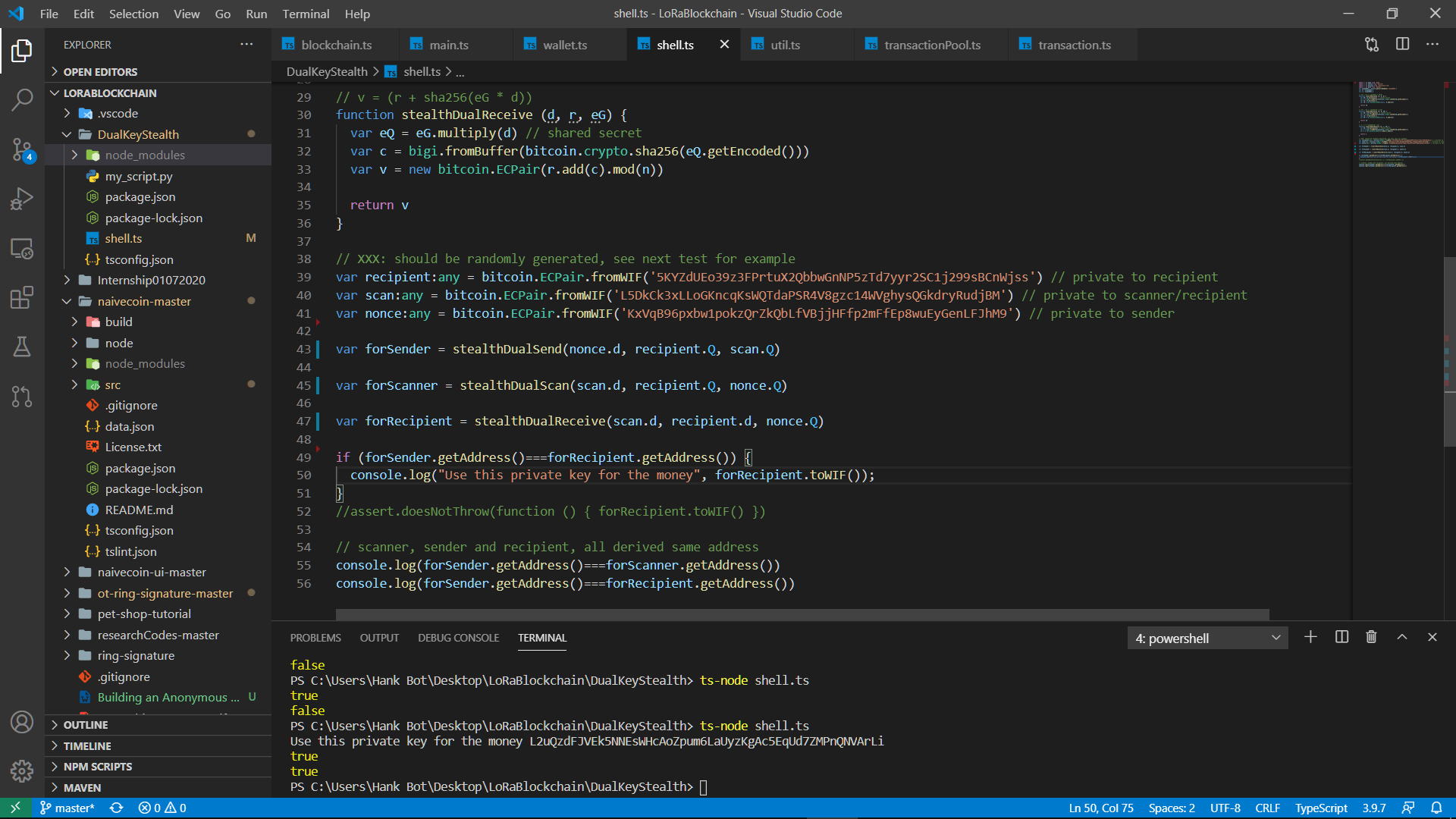


Figure 18: DKSAP Example

Based on Figure 19, both sender and receiver’s stealth address can be verified as the sender’s Q is announced on the blockchain, hence it can be used as a filter for the user’s unspenttxouts for pedersen commitment checking.

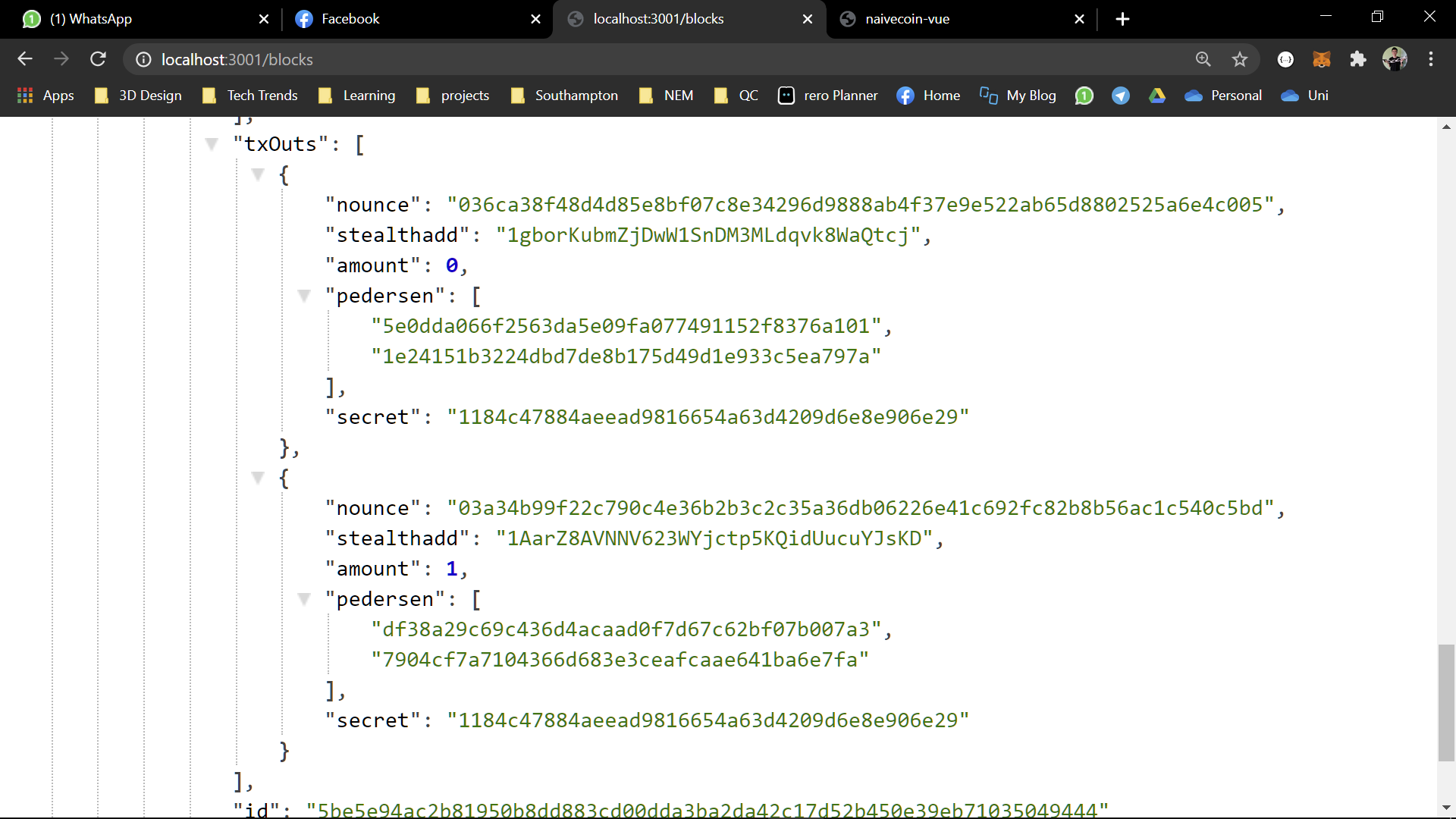
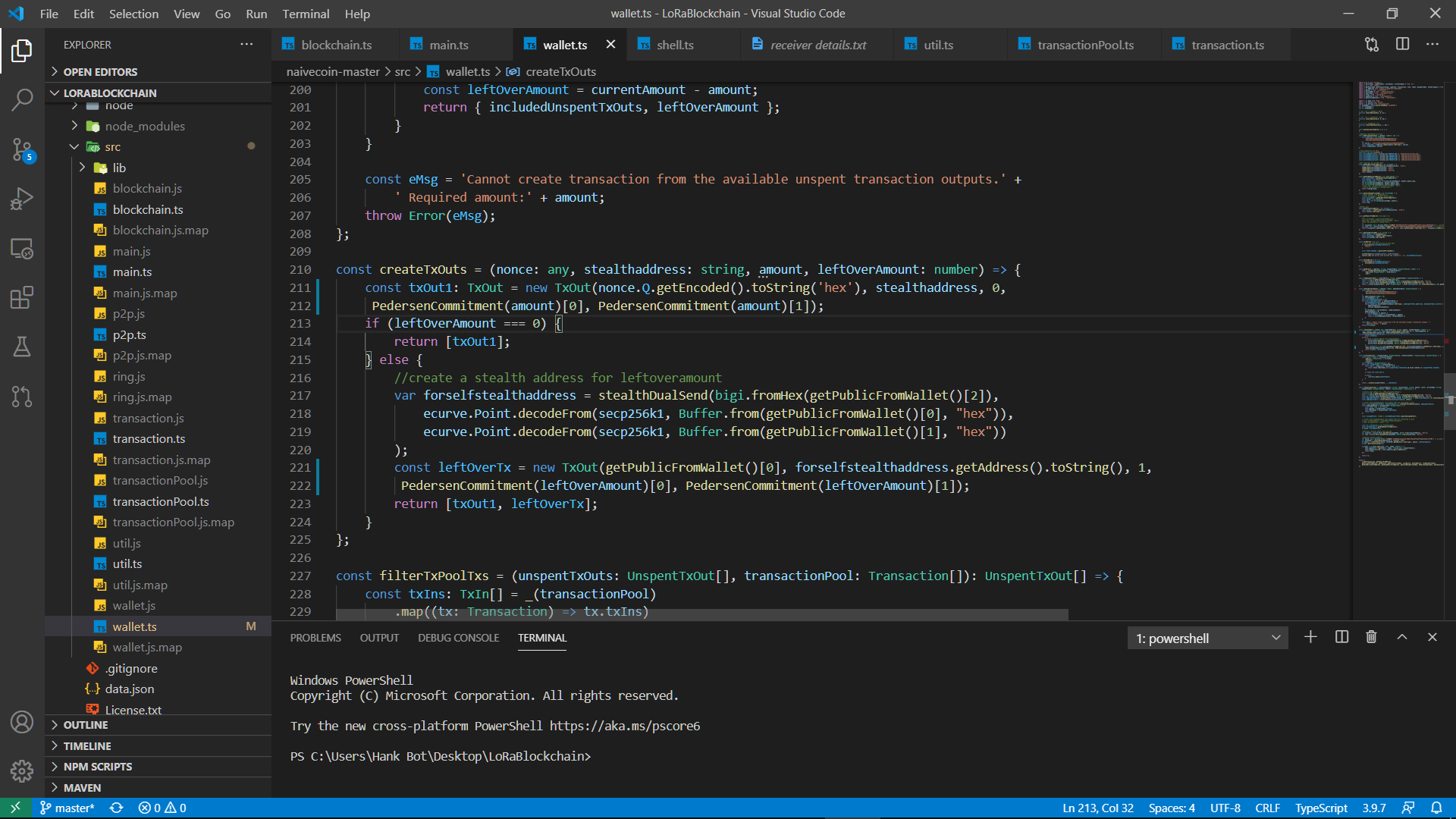


Figure 19: Transactions with Stealth Address

The code for stealth address and pedersen commitments can be found in wallet.ts, function createTxOuts(), where the txOut1 is the stealth address to the recipient, while a new txout is generated for the leftovertx, where the sender and receiver are the same user.



3. Conclusion

References:

Author’s Github Code: <https://github.com/HankRobot/LoRaBlockchain>

Building a basic blockchain (Not anonymous) : <https://lhartikk.github.io/>

Pedersen Commitment Github Code: <https://github.com/Azero123/simple-js-pedersen-commitment>

Ring Signature Github Code: <https://github.com/mtsalenc/ot-ring-signature>

Stealth Address: <https://hackernoon.com/blockchain-privacy-enhancing-technology-series-stealth-address-i-c8a3eb4e4e43>

Monero’s blockchain :  <https://pdfs.semanticscholar.org/16a8/c0aa45bd09830b8c5115c2c1e441f177fc82.pdf>