**The Knuth, the Whole Knuth, & Nothing but the Knuth**

**Bitcoin Blockchain Explorer**

Kody Low: [kody@seas.upenn.edu](mailto:kody@seas.upenn.edu)

David Kim: [dkimhw90@seas.upenn.edu](mailto:dkimhw90@seas.upenn.edu)

Eric Bosma: [ebosma@seas.upenn.edu](mailto:ebosma@seas.upenn.edu)

Farhad Zmarai: [zmarai@seas.upenn.edu](mailto:zmarai@seas.upenn.edu)

1. Introduction

Frequent cryptocurrency traders and users often use a blockchain explorer to check the status of their transactions and gain insight into the activity within the blockchain. They also generate information about wallet addresses, including the assets sent/received and the total value of asset holding. Some users, for instance, like to follow known whale addresses to see where big money is moving.

Bitcoin blockchain data is a publicly available dataset that anyone can download. However, this dataset is not easily accessible or organized in a manner that allows users to easily visualize and explore the blockchain data. Bitcoin blockchain does have ways to pull block and transaction data via `getblock` RPC API calls, but it does not provide easy ways to search for transactions—It can only pull block data by the hash provided to the API call.

The main goal of this project is to build a web application which:

1. Makes it easy for users to check transactions and addresses, and explore block- and transaction-level data
2. Provides major Bitcoin usage metrics in an easy-to-use summary page for users
3. Allows for chain analysis—Track transactions and see whether those transactions made it to an exchange wallet
4. Architecture

Technologies Used

* Data Parsing
  + Python and associated libraries: Psycopg2, Requests, Pandas, SQLAlchemy
  + Esplora API
* Frontend
  + React
  + Material UI
  + Recharts
  + Crypto-js
* Backend
  + Postgres
  + Express

System Architecture

Our web app uses React, a JavaScript library for building user interfaces. We use TypeScript, a typed superset of JavaScript that compiles to plain JavaScript, rather than plain JavaScript to realize a number of advantages: Enhanced code readability, easier debugging and refactoring, and improved code IntelliSense in most IDEs. We chose to use the Material UI library to add several complex components to the site.

Core data is stored in two tables in a relational Postgres database, hosted on AWS. Given the impracticality of storing all possible data about the BTC blockchain in our hosted database, our app supplements the core stored data with additional details from the Esplora API. Backend routing is managed using the Express.js library.

1. Data
2. Database

Relational Schema *(see Appendix A for ER Diagram)*

**block\_headers**(hash, height, version, prev\_block\_hash, merkle\_root, timestamp, median\_time, bits, nonce, size, weight, num\_tx, difficulty, confirmations)

* *Includes 749,471+ tuples (web app posts additional entries)*

**coinbase\_txs**(txid, block\_hash, version, locktime, size, weight, fee, outputs, btc\_mined)

* *Includes 745,524 tuples*

**txidaddress**(txid, address)

* *Includes 478,005 tuples*

Data Ingestion

Given that there are millions and millions of bitcoin transactions, it was technically infeasible to store all transaction data in our Postgres database. Our group chose to combine the block header dataset with coinbase transactions (these are transactions where bitcoin is awarded to miners who validate transactions) to provide the bulk of the data for our web application and use API calls to Esplora to supplement our block level data with transaction data based on user requests. Additionally, we created a ‘txidadress’ dataset to allow easy access to all the transaction ids associated with known exchange addresses. This dataset gives us easy access to look up address data by transaction id or search by address to find all transaction ids associated with a known exchange address.

For all three main tables, we used Python & its associated libraries (Psycopg2, Requests, Pandas, SQLAlchemy) to get block, coinbase, and address data from the Esplora API and insert the data into our Postgres table. To keep the ‘block\_header’ data fresh, when users request the block page, we will check our database’s current max block and compare against Esplora’s max block height. If there is a difference, while the user browses our web app the post request will insert new block data from the Esplora API.

Normal Form

1. Web App Description

*Nav Bar*

Provides quick links to the Home, Blocks, KYC Addresses, and Coinbase TXs pages. On small screen, nav bar links responsively condense into a hamburger menu. A search bar allows users to search by Block ID, Address, TXID, or Blockhash and jump to the appropriate details page (if valid), or display an error message (if invalid).

*Home Page*

The default landing page when the user opens the app, which displays summary statistics about the Bitcoin blockchain. The statistics are filterable over a user-specified date range, and include:

* Average weight per block
* Average # of transactions per block
* Average difficulty per block
* A line graph representation of total transactions by month
* A line graph representation of difficulty by month
* A line graph representation of total BTC mined per year

*Blocks Page*

Allows the user to search through the most recent block headers to see the details of each block. A left-hand menu displays a list of the most recent block headers with block height and datetime stamps, and the details of the selected block are displayed on the main part of the page. These details include a full breakdown of the block header information, and a paginated list of all transactions in the block at the bottom of the page. This “Transactions in Block” section displays 25 transactions per page, and each transaction can be expanded to view the transaction header details or clicked to navigate directly to the Transaction Details page for that transaction. A clickable clipboard icon allows the user to copy the block hash to clipboard.

*KYC Addresses Page*

Allows the user to search through all KYC Addresses in our database to see the details of each address. A left-hand menu displays a list of all addresses, and the details of the selected address are displayed on the main part of the page. These details include a full breakdown of the address details, and a paginated list of all transactions involving that address at the bottom of the page. This “Address Transactions” section displays 25 transactions per page, and each transaction can be expanded to view the transaction header details or clicked to navigate directly to the Transaction Details page for that transaction. A clickable clipboard icon allows the user to copy the address to clipboard.

*Coinbase TXs Page*

*Block Details Page*

Displays the full details of a single block. These include a full breakdown of the block header information, and a paginated list of all transactions in the block at the bottom of the page. This “Transactions in Block” section displays 25 transactions per page, and each transaction can be expanded to view the transaction header details or clicked to navigate directly to the Transaction Details page for that transaction. A clickable clipboard icon allows the user to copy the block hash to clipboard.

*Transaction Details Page*

*Address Details Page*

Displays the full details of a single address. These include a table with data on how much BTC and how many transactions have come into and out of that address, and list of the 25 most recent transactions into or out of that address. This “Recent Transactions on Address” shows allows each transaction to be expanded to view the transaction header details or clicked to navigate directly to the Transaction Details page for that transaction. A clickable clipboard icon allows the user to copy the address to clipboard.

1. API Specification
2. Queries
3. Performance Evaluation

We implemented several optimizations which lead to measurably improved performance:

* Optimization 1: Add B+ Tree index on height in block\_headers table
* Optimization 2: Add B+ Tree index on (timestamp, num\_tx) in block\_headers table
* Optimization 3: Add B+ Tree index on (timestamp, difficulty) in block\_headers table
* Optimization 4: Extracted “btc\_mined” from outputs to create separate column; added B+ Tree indexes on (block\_hash, btc\_mined) and (hash, timestamp) in block\_headers table

1. Technical Challenges

* Too much data to feasibly store
* Implementing Search in Nav Bar—Many possible inputs of varying lengths; how to determine type of input; how to handle invalid inputs
* Kody—Was there some issue with deprecated libraries, or needed to use a previous version of React?
* David or Kody—Anything else to add?