The Difference Between TrueBlocks and *Rotki* and TrueBlocks and *theGraph*

We get two frequent questions about TrueBlocks:

a) How is TrueBlocks different to *Rotki*?
b) How is TrueBlocks different to *theGraph*?

While these questions may seem contradictory, given the differing nature of *Rotki* and *theGraph*, they actually make sense given the nature of TrueBlocks.

In this document, I will go into depth explaining the differences between TrueBlocks and *theGraph*. This will give us a context to explain the differences between TrueBlocks and *Rotki* in a few short concluding paragraphs.

The Difference Between TrueBlocks and *theGraph*

First, I'll discuss the similarities between TrueBlocks and *theGraph*:

Both systems index the Ethereum blockchain

Next, I will discuss the differences between the two systems.

theGraph is an indexing solution for blockchains in general and Ethereum in particular. While *theGraph* is capable of indexing any data, its primary use case is to index events generated by smart contracts for use by the smart contract's developers. *theGraph* is fast and reliable. It can answer queries at web 2.0 speeds primarily because, under its crypto-economic veneer, it is a web 2.0 based API. *theGraph's* "Indexers" (those who produce the index) are paid with an ERC20 token called GRT. This payment incentivizes the indexers to provide a service to distributed applications (dApps). GRT tokens are acquired, by those able and willing to pay, on the open market. There is an extensive (and we think overly-complicated) arbitration mechanism to ensure the accuracy of the result and resolve conflicts. To date, much of *theGraph's* traffic runs through a traditional web-2.0 web server called "The Hosted Service" which is provided by *theGraph* free of charge.

TrueBlocks is also an indexing solution for blockchains (Ethereum only, but extensible). Unlike *theGraph*, which focuses on smart contract events, TrueBlocks focuses what we call "every appearance of every address anywhere on the chain." TrueBlocks is very inexpensive to run, therefore, the incentives to run the software are different. There is no need to purchase tokens on an open market. The software is lightweight enough to be run locally. Finally, the results of TrueBlocks' indexing process are provably-true and distributed web 3.0 native way that

eliminates the possibility of capture, therefore, there is no complicated arbitration mechanism. We explain each of these differences below.

TrueBlocks is a Better Indexer

A simple example of the higher quality indexing provided by TrueBlocks is that *theGraph* does not index "errored" transactions. This is due to the simple fact that events are not generated for transactions that end in error. TrueBlocks indexes every appearance of every address – both in-error and not in-error. While things may have changed recently, the last time we looked *theGraph's* own documentation still notes that only event-related results are generated.

Another example of the superiority of TrueBlocks indexing is shown by the following anecdote.

About a year ago, we did a study on the top forty ERC tokens. We looked, as we will, at "every appearance of the addresses anywhere on the chain." We used EtherScan (not *theGraph*, but similar results are expected) as the source to establish a base case. EtherScan has five primary API endpoints related to transactional history (mining, transactions, internal transactions, token transfers, etc.). We wrote scripts to query all five endpoints, remove cross-endpoint duplicates, and create a data set that represented "every appearance of the token according to EtherScan."

We ran the same extraction using TrueBlocks.

For all forty tokens, TrueBlocks found more appearances than EtherScan. In fact, the ratio was consistently around 3:2 in favor of TrueBlocks. In other words, if EtherScan found 200 appearances for an address, TrueBlocks found 300. This is because of the way TrueBlocks scans the data (including traces, events, contract-creations, self-destructs, and importantly arbitrary byte-data). We've explained this elsewhere on our blog.

We found further that, in almost every case, it was because of these "missing" appearances that transactional histories produced from EtherScan data does not reconcile (in the traditional accounting sense). Ask any crypto tax expert how easy it is to reconcile an Ethereum account. They will tell you it's impossible. With TrueBlocks, it is possible – we will return to this when comparing ourselves against *Rotki*).

We suspect the *theGraph* has similar shortcomings in its indexing behavior as EtherScan (probably worse as they are focused only on events).

In summary, TrueBlocks is a better indexer than both EtherScan and *theGraph*.

TrueBlocks is not a Market-Based System

We run our own Ethereum node. About three years ago, we built a custom-made Linux box for about $2,500 US. We've been running OpenEthereum on it ever since. We also run the TrueBlocks indexer on that same machine. The machine easily keeps up with the front of the chain. In fact, the machine is almost always resting, coming alive only to handle new blocks once every 13 seconds. The cost of running this machine, in house, is near zero.

Periodically, TrueBlocks "consolidates" a chunk of its ever-growing index of appearances creating what we call a "time ordered log of an index of a time ordered log." Ethereum is, at its heart, a time-ordered log of transactions. TrueBlocks indexes that time-ordered log, but it periodically consolidates the index into chunks. That is, it produces a time-ordered log of the index chunks.

Because the index chunks are time-ordered they are immutable. Because of that, we can write the indexes onto a content-addressable data store such as IPFS. We do this as each chunk becomes available. We also produce a bloom filter at the same time that is also immutable. The bloom filter is orders of magnitude smaller than the chunk. The cost of producing the chunks, creating the bloom filters, and publishing both (and pinning both) to IPFS is near zero. (We use a pinning service called Pinata for this.)

The final step in publishing the TrueBlocks index of appearances is to collect the entire list of IPFS hashes in a manifest which we then also publish to IPFS. With the manifest, one may recreate the entire index. That is, a single 32-byte IPFS hash represents the entire index of every address anywhere on the chain.

Daily, so as to lower costs, we publish that 32-byte IPFS hash to a smart contract we call "The Unchained Index".

Our frontend software, which we will discuss when comparing ourselves to *Rotki*, then reads the IPFS location from the smart contract and downloads the bloom filters. When searching for a list of transactions on a given address, we use the blooms decide which parts of the full index to download from IPFS. The costs of publishing to the smart contract is dependent on the cost of gas, but the smart contract was purposefully made very simple so as to minimize these costs.

We estimate the total cost of publishing the index, daily, to be less than $20,000 US annually.

It is very important to note that the TrueBlocks index does not require any delivery mechanism. If an end user has access to an Ethereum RPC endpoint and access to IPFS, they may acquire the index without reservation. Once we publish the hash of the manifest to Unchained Index, it is impossible for us to take it back.

Furthermore, unlike web 2.0 data delivery, our system gets better and costs less the more people use it. This is because more copies are being pinned as download and pin themselves their own portion of the index. This behavior is a natural by-product of a content-addressable store which we take advantage of by design.

A further interesting aspect of the TrueBlocks way of data distribution is that each user downloads only that portion of the index that they are interested in. A person who uses the blockchain minimally may only download a few chunks. While another person, who interacts more frequently with the chain, and therefore appears more frequently in the history of the chain, will download proportionally more chunks. Each user pins their own chunks (because they want to ensure that the chunks are available for their own use). The system naturally distributes the index to users based on their own usage. Heavy users download and pin a larger portion of the index. Light users download significantly less. This is a naturally fair system with near-zero cost.

All of the above is in contrast to *theGraph's* open market. That system is designed for users to compete for the service provider's services. This will lead, obviously, to those with a higher tolerance for high prices (the rich) to focus the indexers limited resources on their queries. This will (we believe), preclude those more sensitive to price increases to be unable to get their own transactional history.

In summary, because we've purposefully designed TrueBlocks to produce index data and publish at near-zero cost, we can effectively share that data without creating a competition among users. Additionally, the more people use the data, the cheaper and more readily available it becomes. This takes perfect advantage of the network effects provided by content addressable data. In fact, the estimated $20,000 cost of producing the data and publishing it to the smart contract will be borne by us as a very small part of our cost of doing business.

TrueBlocks is Provably True

This is probably the biggest difference between TrueBlocks and *theGraph*. The index that TrueBlocks creates is immutable, can be verified once and used indefinitely, and can be shown to be both reproducible and provably true.

We accomplish this with a very simple trick.

As we "consolidate" each chunk of the index and prior to storing it onto IPFS, we insert a special tag into the data that provides enough information that anyone would need to reproduce the results of the calculation. That tag comes from a very simple to find place: `git log | head -1`. In other words, we insert the git commit hash of the code that produced the chunk into the chunk. This means anyone obtaining the chunk can (if they wish) reproduce the data simply by rescanning the chunk using the code at that git commit. This aspect of our work is somewhat of a work in progress and more details will follow.

Unlike *theGraph* which requires a very complicated, expensive, and we believe prone to failure arbitration court (which, by the way, *theGraph* stands in the middle of on the final word), our system is automated. The commit hash is inserted during the regular operation of the indexing task.

Furthermore, the data TrueBlocks produces (like blockchain data) is immutable. This means that any group of people can convince themselves that trail of hashes published to the Unchained Index is correct and the veracity of the data need never be questioned again. We call this audit-once data. It falls out of the back of the system as a by-product of chunking and publishing the index portions to IPFS.

Summary of Comparison of TrueBlocks with theGraph

Every one of the design choices we've made about TrueBlocks is focused on producing a near-zero cost, provably true index of every appearance of any address anywhere on the chain. This index allows us to build a different breed of decentralized desktop applications that we call Web 3.0 native. We've even envisioned eliminating the standard web 2.0 software stack as all data (including the source – the Ethereum node) is local. There is no need for over-the-wire data delivery protocols. We've written about these aspects of our project extensively on our blog (http://docs.trueblocks.io/blog).

We are now in a position to compare ourselves with Rotki, which we will do in just a few more paragraphs.

Comparison with *Rotki*

Rotki is a python-based desktop software application devoted to preserving end user privacy and giving the user a look into his/her own current blockchain related holdings. It makes extensive use of third-party APIs which enable it to report on many aspects of a user's daily blockchain usage including holding not only on the various Ethereum chain, but other chains as well, including Bitcoin and others (sorry – I don't have a list).

The Rotki software is a sometimes user of EtherScan, a sometimes user of theGraph, a sometimes users of feed for coin prices, etc. It is also a sometimes user of the Ethereum mainnet RPC. Having had conversations with the developer of Rotki (Lefteris), we are aware that they have a pretty difficult time using the Ethereum RPC to acquire historical lists of transactions on a user's account. For this reason, they resort to using EtherScan to produce such lists. This has all the problems mentioned above.

Rotki cannot complete what, in the traditional accounting sense, would be called a reconciliation of the accounts under its management. This is because its sources of data (EtherScan) are inadequate as we pointed out above. For this reason, Rotki does not market itself as an accounting solution for blockchains – only a portfolio manager. Providing front-of-chain balances for an account is easy with an Ethereum endpoint. Providing reconciled, 18-decimal-place accurate account is impossible.

Because TrueBlocks can (through its backend) acquire “everything that ever happened on an address”, the TrueBlocks Account Explorer (http://github.com/TrueBlocks/trueblocks-explorer) can reconcile every transaction on an account. This is basically the accounting holy-grail. Instantaneous, every-13-second, 18-decimal-place-accurate reconciliations of an account.

Another aspect where *Rotki* and *TrueBlocks Account Explorer* differ is in its ability to find and detail ERC20 token holdings. The *Rotki* software keeps a list of known ERC20 token contracts. When presenting a list of token holdings, the software consults this list and, after pricing by consulting a third-party price feed, presents the holding for the users perusal.

In TrueBlocks, on the other hand, because we have a list of “every appearance of the address on the chain”, we can look at the transaction itself. If we suspect the transaction is token related (this is fairly easy by looking at the four-byte signatures) or if we’ve previously determined that the address is a token, we can query the token directly for the address’s balance. We also have a mechanism for consulting UniSwap for the price of a given token relative to ETH and ETH relative to DAI to price in US dollars. In this way, we can price every transaction at its moment of inception. We do not have to fall back to daily pricing as many blockchain tax consultant would. This is a direct results of the way the TrueBlock backend works (TrueBlocks Core).

One further very difficult to explain aspect of TrueBlocks that is not available anywhere in the world that we are aware of is something we call “User Centric Browsing”. This is the idea that, because we have a list of everything that ever happened to an address, we can allow the user to scroll through his own history (or anyone else’s for that matter). This has the very odd experience of seeing right after each other a transaction with ENS, and then a transaction on the same day with UniSwap, then theGraph, then Giveth. All of these transactions are being presented in one view. The user need not go to ENS to see his/her ENS transactions. He need not go to a DAO to see his/her DAO interactions. Everything is directly in front of the user in a consolidated view – a user centric view.

Summary of Comparison with Rotki

TrueBlocks and Rotki are similar. Both are desktop software. Both want to give the user a perfectly private experience without website snooping. Both want to allow the user to understand his/her own accounts.

Rotki is more broadly focused on other chains, but that comes at the cost of not being fully decentralized. Rotki depends heavily on third-party websites for various sources of data.

The TrueBlocks Account Explorer is 100% decentralized because it is based on the index produced by its own backend (the Core). This means that the TrueBlocks Explorer can run directly against the Ethereum node to acquire its data. This has positives and negatives. It provides perfect privacy. Nearly the entire system gets its data directly from the node. However, because the Ethereum database is honed for delivering block data to the network, its

access is a bit slower than a regular database. In order to work on small-machine desktops, we trade off some slowness for full decentralization. We do have a very heavy binary caching mechanism, so the software is only slower on initial query, but it does behave different to a website.

Summary

I'm pretty sure I've gone on long enough. In fact, I'm pretty sure I've gone on way too long, but I'm trying to help the reader understand that TrueBlocks lives somewhere half-way between theGraph and Rotki. TrueBlocks can do everything *Rotki* can do only better and fully decentralized BECAUSE it can also do everything *theGraph* can do (only better and significantly cheaper and provably so).

Our work is self-funded with the exception of a 2018 grant from the Ethereum foundation ($120,000 US) and 2019-2020 grant from Consensys ($25,000). We look forward to further conversation in relation to our request for a grant for your fine organization.

Cheers.
Thomas Rush
Philadelphia, June 8, 2020