Partitioning Bitcoin UTXOS Merkle Tree Into Different SubTrees To Reduce Proof Size

Abstract. The notion of Stateless nodes has emerged in Blockchains as an acceptable solution to security and scalability constraints. Stateless nodes store an accumulated hash value, the root of a Merkle Tree, and fetch proofs as needed to verify the current block; as opposed to full nodes that keep a full copy of the system state and light or pruned nodes that rely on full nodes to verify. In Bitcoin, the system state is the current set of all Unspent Transaction Outputs UTXOS, and the proof of a leaf UTXO is the hashes of its siblings along its path to the root. In this paper we propose a straight forward improvement on the proof size by simply applying the divide-and-conquer approach; ie, divide the UTXOS set into different partitions, each to be Merkleized separately. Whatever the partitioning criteria is, a smaller size problem will be solved by parsing only the corresponding subtree; unlike most sharding solutions, no cross shards coordination is needed here, even if UTXOS of one transaction falls in different subtrees each proof is fetched separately. Partitions could be chosen to categorize the UTXO set according to semantic classifiers. We explore and analyze the merits of different criteria; we consider UTXOS with probability almost zero of being spent such as non-standard, burned UTXOS, and dust UTXOS, also coinbase UTXOS and their different spending pattern. Then we suggest a promising economic classifier, where UTXOS are partitioned according to their BTC value. Finally, we discuss performance and security impact of our proposal, and close with some related research points as a possible future work.

Keywords: UTXO, Bitcoin, Merkle Tree, Stateless Nodes, divide-and-conquer, scalability.

1 Introduction

Blockchains could be viewed as an append only data structure, with its size increasing endlessly as more blocks are added. In Bitcoin, full nodes, that form the consensus, keep the whole chain in its secondary storage and the whole UTXOS set (~4G) in its local memory in order to correctly verify transactions in each block. Lightning nodes and SPV clients on the otherhand keep only block headers and depend on full nodes to verify which makes them more vulnerable to attacks. Stateless nodes is a reasonable compromise where nodes keep an accumulated hash value in their local memory and fetch a ***proof (witness)*** to verify UTXOS in the given block. UTXOS Merkle Tree works the same way as transactions Merkle Tree whose root is stored in the block header [1]; hashes of UTXOS are inserted as leaves and a tree is built on them, the hash of a leaf is calculated by extracting sibling hashes along the path from the stored accumulated root value and then compared with the given UTXO hash[2].

This paper is concerned with Bitcoin; stateless nodes concept, however, is common to all types of cryptocurrency blockchains. In general there are two main types account-based like Ethereum and Unspent Transaction Output

UTXO-based like Bitcoin; example of two-mode cryptocurrency is Cardano[3].

Account-based Blockchains follow the regular banking, and programming, paradigm; data is structured according to created accounts, that act like variables storing the amount of currency (account balance) and are updated normally through transactions like in banks or variables through lines of code. Ethereum is the most famous account-based cryptocurrency, where the complete state-trie has to be kept by a full node; aiming to reduce proofs their researchers used a Merkle Trie that evolved several times[4], and most recently switched to polynomial commitments[5].

In ***Unspent Transaction Output based Blockchains***, like Bitcoin, coins or currency is the main component the system has to keep track of[6]; ie unlike the normal banking model a transaction output UTXO is only used once as an input to a new transaction then its lifespan ends (becomes spent). The concept aims to prevent double spending attempts, where any money transfer on the ledger is viewed as the end of existence of the transaction inputs (that was before an Unspent Output of a previous transaction) and hence the creation of new Unspent outputs tied to this transaction. The state of the blockchain at any instance(block) is defined as the current set of Unspent Transaction Outputs UTXOS. An example other than Bitcoin is the UTXO-based mode in Cardano, where the UTXOS set is divided into four types, creation is restricted to only one UTXO per transaction, AVL+ tree is the underlying Merkle structure, and there is a special dust handling [3].

In Bitcoin, the problem of the growing size of the UTXOS set have been a worry [7,8], researchers tried to reduce proof sizes by choosing a balanced data structure for the Merkle to reduce the leaf-to-root path length [9[1],10]. The main heuristic is similar to that of multi-level cache "outputs that were spent recently are more likely to be re-spent (the new UTXOS resulting from them) than outputs that have not been spent in a long time" [6], the ***locality of reference*** was mentioned in [10]. However, they all treated the UTXOS as a whole set, we didn't find any previous partitioning attempts in the literature. We also didn't encounter the use of econmy and market metrics that we explore here; spending behavior is described as *spending waves,* where long lifespans represents long time holders ***LTH*** that appears in ***HODL waves*** related to BTC price. We then suggest to partition the UTXO set according to BTC value. We also present guidelines to recognize & separately store some types of UTXOs that have different spending patterns than the common mainstream, types that moves together according to market analysis, and types that are unspendable or have almost zero probability of being spent; an approach that will definitely reduce the problem size.

The rest of the paper is organized as follows section2 gives a brief survey of data structures suggested for storing UTXOs hashes ending with the Utreexo project. Then section3 starts with research efforts in analalyzing UTXOS spending behavior, moving to our share of the analysis, till we conclude some suggested UTXO types to be Merkleized separately; more details are in

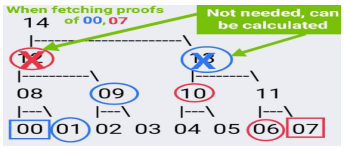
[1] several kinds of non hash based accumulators (do not use a Merkle strucutre) were only mentioned in [9] with the drawback of their excessive arithmetic which could be time consuming and that they are not post quantum secure. However, they may resemble the newer Ethereum approach in [5].

Appindex A. Section4 explores some market metrics and partitions that could be derived based on them. Hence follows analysis of our defferent proposals in section 5. Finally section6 concludes the paper with plans for future work.

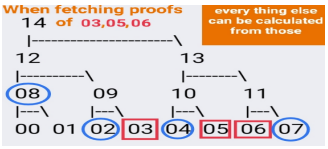
2 Data structures Used for UTXOS Merkleization

As early as 2011, both Greg Maxwell [11] and Dithi [7] suggested using a Merkle tree in order to query the UTXO set more efficiently. This was followed by recommending a TX-id indexed Merkle prefix Trie in 2012[12], and Red-Black trees in [13]. Ever since then one can find a considerable number of projects trying to suggest different kinds of trees or data structures to store the hashes; the accumulator batching techniques introduced in [14] could be viewed as the end of those using the hash as the BST key of the Merkle strucutre. The accumulator, which was not specifically design for the UTXOS case, has superior properties in terms of proof size and batch reductions; however, sometimes batching all required UTXOS proofs in a block is not valid. In Fig. 1 for example, you must recalculate the root between fetching node 00& node 07 in (1-a), or nodes 03,05,06 in (1-b); which changes the internal nodes values and make the reduction invalid.

Another research series chose to sacrifice the *proof of non-inclusion*[2] feature and just append new UTXOS to the right to promote a *locality of reference* heuristic that achieved more inner node reductions in batch proofs. An NSF funded research suggested the use of a Merkle Trie in 2021[9] gives a detailed discussion of such heuristic; when UTXOS are appended to the right UTXOS spent in the required block becomes adjacent or at least clustered which results in more common sibling internal nodes in the fetched proofs. Examples of those reductions and how the leaves adjacency leads to even more are illustrated in Fig.1 a,b; the figures are inspired by the comparisons in [15], although not in the context of cryptocurrencies it views a set of required proofs as *a sparse tree* taken from the complete tree of all possible leaves.



a) 4nodes needed for 2 apart nodes.



b) still 4nodes for 3 near nodes

Fig.1 examples of batch proof reductions when some siblings can be calculated

Merkle Mountain Range MMR [16] was the first to our knowledge that dropped the hash keyed tradition and appended new UTXOS to the right. One could say that they originated as early as 2016[8] the intuition of using a number of *complete binary trees*, ***mountains***, that reflects the *1s* in the binary representation of the number of leaves(UTXOS) with depth equal to their position. For example a set of say 11 UTXOS = $(1011)_2$ can be stored in three complete Merkle binary trees of depth 3,1,0; another hidden advantage of

[2] To prove a UTXO doesn't exist, in hash-keyed Merkle strucutres, two proofs can be submitted that the nearest UTXO hashes before and after the missing one are adjacent leaves. If the Merkle is complete with a default NULL hash value for non existing leaves, like in [15], one proof is sent that leads to the NULL hash to be extracted.

complete binary trees is that they can be stored compactly as vectors without the need to use left and right pointers. MMR is available as API and are now used in *Tari* cryptocurrency project[17].

On the same track followed **Utreexo**[10,18], an MIT dci-lab project that suggested ***a forest*** of complete binary trees stored in a bridge server, and new UTXOS get appended to the right. Specifically, new leaves start building smaller subtrees in powers of 2 that only joins the next larger subtree (next power of 2) when it becomes full. This guarantees a *bounded tree depth* of ***log(no of UTXOS)*** , and hence number of siblings in fetched proofs, with adding only ***log(no of UTXOS)*** *accumulated roots.* The designers say [20] an average block (about 2K transactions) with a few thousands of insertions & deletions only causes something on the order of 10K hashes, which is *much less than logN* per element (26-27 for the current UTXO set size of 78m) due to the *locality of reference* and the recent spending heuristics where most of the activities happen in the smaller powers of2 trees; the swapless implementation of the delete operation the team say they added in 2021 may have contributed to this. Utreexo also studied the cache deployment for the fetched proofs & IBD Initial Block Download, and ***added block number to the UTXO hash*** to increase security and prevent pre-image attacks.

3 Analyzing the UTXOS Spending Patterns

Recent results in [9,10] described the lifespans of UTXOS as a Pareto Distribution; a fast decaying curve with mainstream majority at length of 1-2 blocks. Although the Utreexo paper mentioned the existence of peak at age 100 due to block confirmation time (miners can't spend their mining reward UTXOS, ***coinbase UTXOS***, before 101 blocks), this was not enough to explain our observations through some sampled data, especially coinbase UTXOS. From a quick sampled coinbase data set, 101 was only the minimum, but not average or median; Table1&Fig.2 summarize the results. We also noticed they are usually spent together in a fourtune gathering transaction by miners, or a fortune splitting transaction by a pool; an observation that is consistent with[26], and will add more locality of reference if they were stored together.

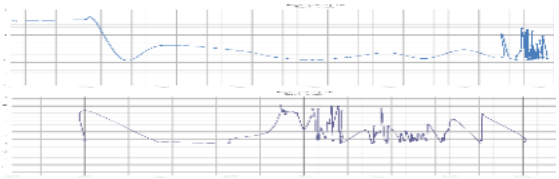


Fig.2 Lifespan of sampled coinbase UTXOS drawn at logarithmic scale

Table 1: coinbase UTXOS Lifespan sampling results

Block range	All	≥150,000	≥500,000	≥650,000	≥680,000
Min	101	101	105	105	105
Average	16518.9	3251.69	3453.55	3566.91	2656.97
Median	525.5	455.5	465	467	448
Max	167300	33139	33139	33139	26105
#ofSamples	100	90	84	81	67

So, we had to search deeper for more studies analyzing the UTXO set [22,23,24,25]. We found a study [21] where coinbase UTXOS had almost an opposite pattern to that of the whole UTXOS set[3]. The paper mentions some striking numbers, although outdated to Feb2018, more than 50% of UTXOS, exactly 37,728,355 UTXO, had a lifespan more than 6 months (28,456 block); the data was measured accurately due to the Bitcoin cash fork in Aug2017.

In [22] the numbers in the Stanford report (Dec2015) shows that only 44% of the UTXOS could be in thatvery short lifespan interval of the Pareto Distribution (Fig.3a). Although the final result of the report described the UTXOS set to be *Chaotic* and cannot be fitted into a distribution, the report suggested the idea of adding multiple Laplace distributions, which accommodate with our intuition that the UTXOS set is better described as composite from different types; Fig.3b shows how the Histograms are different for each cluster. Recently, [24] used *Cohort analysis* to interpret Bitcoin data using methods developed for population data in social science.

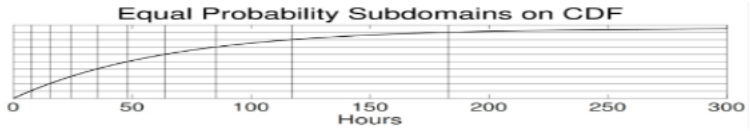


Fig.3 (a)vertical lines represents equiprobable intervals of 10% of UTXOS

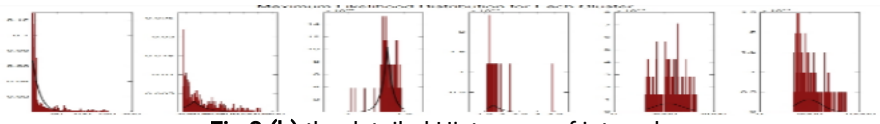


Fig.3 (b) the detailed Histogram of intervals

It is also worth mentioning that the removal of *age priority* from the Bitcoin Core1.15 coin selection in 2017 [23], may have to the more longer lifespan ratio in 2018 data[21] than that of the Stanford report. To be accurate it contributes more to Bitcoin main heuristic that outputs created recently are more likely to be spent than elder ones (since the favoring of old UTXOS as inputs, or any FIFO approach no longer exists). Also, the fact that very earlier blocks didn't contain transactions, and the number of transactions per block increased with time, make the ratio of coinbase UTXOS (must have age ≥101), and thus their contribution to the average age of the whole set, less with time.

Now we go back to the Pareto curve presented in the Utreexo paper to see how it can explain the above results of longer lifespans; we find that a closer look into the curve can give a different interpretation of the values (Fig.4). The curve shows that in 2019, when the number of UTXOS was about 65m, about I(10^7) has a lifespan of O(10^0), still there are 10^5 with life of order 10^3, 10^3 with life of order 10^4,... *these are still considerable numbers in the overall performance*. Also, the thickness at the very large numbers cannot be ignored, if these are very adjacent values then they should have been estimated with an order number that sums all values greater than say 10^5. We believe these numbers do affect the performance, only the forest design in Utreexo

[3] Figures from [21] are in Appendix A; more figures & analysis on our sampled data in Appindex B.

diminishes the effect of any increase in the old UTXOS ratio; since they remain in the larger most left subtree, which root is probably cached, without affecting the fast dynamicity of the append/delete in right subtrees.

Fig.4 the power-law Pareto Distribution Utreexo curve revisited

We believe that the old become older is not the only heuristic that govern the UTXOS; newly generated dust, burned, and sometimes coinbase UTXOS are also going to stay for very long. These long living UTXOS should not burden the fast movement of the other kinds, and their witness hashes should not be updated with every change either. Whenever ***sharding*** is mentioned it is always accompanied by the burden of needed communication between shards; here we are lucky enough to have the reverse situation, partitioning save us the unnecessary coordination between UTXO subsets. For example, in Utreexo forest those UTXO kinds will be appended at the right as usual and will start moving gradually towards the left larger subtrees, changing internal (non leaf) nodes with every step along the way; we can save these unnecessary changes of possibly cached values if we could find a way to identify long living ones from the beginning. Therefore, our first proposed partitioning is criteria-based; ***we propose to separate 4 different kinds of UTXOS from the mainstream in a separate Merkle Tree for each. Namely, coinbase UTXOS*** [24,25,26] for their different spending pattern***, burned UTXOS*** [27,28,29,30] we already know about 163,226 UTXOS that will never be spent, similarly about 3.5-5m ***dust UTXOS*** [31,32,33,34,35]***, and*** more than 422,822 ***non-standard UTXOS*** [36,37,38,39,40] **.** This separation also helps in identifying/preventing certain attacks that use the nature of those subsets; see Appendix A for more details.

4 Partitioning on market metrics bases

The authors in [41][4] related the UTXOS lifespan median to demand increase, which points out that market factors does affect the UTXOS spending pattern. In fact, data scientists from the market prediction field have done their share of studying UTXOS lifespan too [42,43,44].

In their terminology ***HODLing*** refers to long-term investors deciding to keep their money expecting BTC price increase which means longer lifespans for their UTXOS, and ***liquidation*** means investors deciding to sell ending their UTXOS life; short-term investors are those whose coins keep moving

[4] see figures 5&6 in their paper [41],the earlier shows a linearly increasing median lifespan of some of the UTXOS, with a lot approaching the X-axis also, the latter tries to interpret the linear increase as a reflection to the increase of transactions in memory pools, ie demand increase.

continuously leading to shorter UTXOS lifespans [44]. They set up a threshold of 155 days **LTH** to split the two groups, and yet their curves don't just monitor the two groups continuously; the lifespan range is divided into 10 ranges with their ratios measured [43]. One could think that equal partitions could be tuned from the statistics of those age ranges; however, this would involve a vague costly part of dealing with newly generated UTXOS that will keep moving between age clusters. Yet, the market can introduce to us another partitioning criteria; ***BTC value***.

In this paper we suggest to split UTXOS into nearly equal partitions according to their value ranges.The value classes described in [45] along with their ratios, shrimps, dolphins,...etc and the corresponding UTXOS ratios measured in [46,47] is an easy handling promising partitioning criteria; see Fig.5. It is important to note that we do not mean to use the exact value ranges as partitions, but to use their continuous UTXOS ratios measured by *glassnode* and *bitInfoCharts* to tune divide the value range into expectedly equal partitions that may merge or split the conventional market clusters.

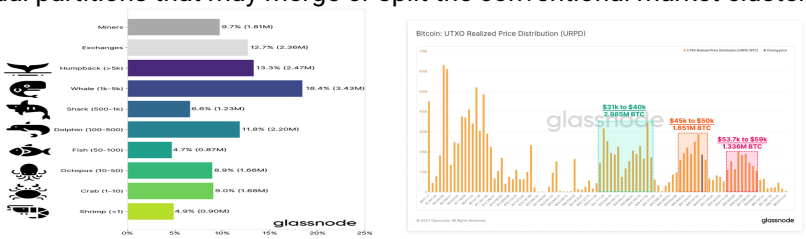


Fig. 5 ratios of BTC values and their corresponding UTXOS

5 Analysis of the proposed scheme

In this section we give a brief analysis of the impact of our proposal from complexity and security perspectives.

5.1 Complexity Analysis

1-Starting from the abstract divide-and-conquer concept, for a problem of size N and a solution of order complexity F(N), if we could figure out a way to either solve a problem of size X, say with probability P, or a problem of size (N-X) with probability (1-P). Then the expected complexity becomes:

P*F(X)+ (1-P)*F(N-X) ≤ F(N) as long as F(N) is monotonically increasing with N

In other words, as long as solving a problem of size **N1<N2** have a complexity **F(N1)≤ F(N2)**, reducing the problem size will reduce the complexity.

For the UTXOS proof size problem discussed in this paper, ***F(N) could represent the proof length in nodes***, it is true that some tree structure could lead to a larger path length for a tree of size N1<N2. However, the excessive number of fetch operations per block (~1000-4000), and the balancing efforts in most used structure are enough to use amortized analysis and assume with a clear conscious that F(N) is monotonically increasing with N. In addition, the burden added for the partitioning is of constant size say C; ie, the roots of the added subtrees are not a function of N, and could be kept in main memory or cache as only one of them changes per operation. ***Hence it follows that, the improvement in the proof size is guaranteed, we are just***

analalyzing its expected ratio. We remind that there will be also a guaranteed reduction in time complexity, although not the dominant factor here.

If **F'(N)=P*F(X)+ (1-P)*F(N-X) + C** , where C is a constant~4-8

It follows that **F'(N) < F(N) amortized**

2-Had we been able to use the market metrics, whether based on HODLing threshold or on BTC value, to tune the partitioning into nearly K equal partitions each of size O(N/K). Say we had 4 partitions it will be like handling a UTXO set of 20m instead of 78m ; the complexity becomes

F'(N)= F(N/K)

since **F(N)=O(log N)**

F'(N)= O(log N/K)

the **improvement ratio = (O(log N)- O(log N/K))/O(log N) = O(log K)/O(log N)**

This means for the current UTXO set size of 78m and say 4-8 partitions ~ **7.6-11.4% reduction in proof length**

3-If we just suffice with the partitions suggested in section3 (detailed in Appindex A) and kept the large set of the mainstream as it is, the dominating complexity will be that of the mainstream UTXOS. As a rough estimate **lowerbound**, we have trimmed 0.542% non standard UTXOS, 0.2092% burned UTXOS, 4.448%-6.362% dust (6.36% addreses holding ≤1$ [47]), and coinbase UTXOS; ie, not less than **~7.11% reduction in the problem size**.

5.2 Robustness/Security Analysis

Since the Partitioning idea could be viewed as an upper layer, just an added step before Merkleization, we believe security is much dependant on the underlying system security than the Partitioning. However, like the block height in Utreexo, the Partitioning criteria is ***an extra semantic piece of information*** the server now posses about each UTXO and can check. For example, an adversary trying to fraud a small value UTXO with a larger BTC value maybe more detectable in a UTXO set bounded by a value range. Also, some specific partitions may provide an attack detection heuristic an; example is repeated dust sending discussed in Appindex A.

6 Conclusions & Future Work

In Blockchains stateless nodes stores only an accumulated hash value of the system state, the UTXOS set in Bitcoin, and fetch proofs needed to verify the current block. In this paper we applied the divide-and-conquer concept and proposed to partition those hashes stored in whatever used Merkle strucutre to reduce the proof size; in addition, this will lead to more locality of reference and allow better handling of each kind with its own spending pattern. We studied UTXOS different kinds and especially coinbase (miner rewards)spending patterns, and devised a preliminary step to remove the burden and prevent suspecious uses of the non-spendable ones. We also recommend to separate coinbase UTXOS for their different spending characteristics. Then, we showed how promising could be BTC value as a splitting criteria that could lead to nearly equal partitions. Finally, we briefly discussed the expected performance and security enhancements; the space reduction could reach **11.4%** if we could tune to 8 nearly equal BTC value

partitions, and in the worstcase of just separating non spendable UTXOS, we can achieve about **7.11%** guaranteed reduction in problem size.

Possible future work plans include ***tuning the best BTC value to equally partition the UTXO set*** with the guide of statistical sites like [46,47], and could also include finding the optimal tree kind for the underlying Merkle fitting each partition spending pattern. We may study ***the applicability of the idea to full nodes***[5] where least probable to being spent UTXOS can remain in secondary storage of full nodes; a problem arises in checking them before declaring a transaction as invalid. However, they still can be excluded from sampling in things like *Bloom-Filters* and assume-UTXOS, or full node cache to increase the probability of a hit. Also could be more appealing if we study ***the idea of wallets being partial full nodes*** by holding say 25% of the UTXOS set and stricting their customers to certain range of UTXOS in return of higher performance. ***Providing Non-inclusion proofs*** is also an appealing challenge that was brought up recently as a criticism for append to right Merkle strucutres. Probably any append to right design share with Utreexo, the use of a mapping strucutre to map hash indexed UTXOS into birth sorted leaves. We think a carefully designed map could be combined effeciently with a secondary hash-keyed Merkle on top of it, so we can provide *fraud proofs w*hen the UTXO is not found.

Acknowledgment

I'm grateful to all those who put their material, especially lectures & papers, free online and to answers & brainstorming in discussion boards and mailing lists. I also owe a lot to my oldest professors in my native faculty, especially my post graduate supervisors who remained in contact and always encouraged me through hard times in life. If this paper is accepted, I dedicate it to the soul of my main supervisor 20yrs ago.

References

1. Russell O'Connor, "A Method for Computing Merkle Roots of Annotated Binary Trees", May 2017, https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2017-May/014362.html
2. Teemu Kanstrén, "Merkle Trees: Concepts and Use Cases", https://medium.com/coinmonks/merkle-trees-concepts-and-use-cases-5da873702318, 16 Feb 2021
3. "Understanding UTXOS in Cardano", Design specification paper, https://emurgo.io/blog/understanding-unspent-transaction-outputs-in-cardano, https://hydra.iohk.io/build/790053/download/1/delegation_design_spec.pdf
4. https://blog.ethereum.org/2020/11/30/the-1x-files-code-merkleization/
5. https://vitalik.ca/general/2021/06/18/verkle.html
6. C. Pérez-Solà, S. Delgado-Segura, G. Navarro-Arribas and J. Herrera-Joancomartí, "Another coin bites the dust: an analysis of dust in UTXO-based cryptocurrencies", January 2019, Royal Society Open Science 6(1):180817 DOI:10.1098/rsos.180817
7. https://en.bitcoin.it/wiki/User:DiThi/MTUT#PROPOSAL:_Merkle_tree_of_unspent_transactions_.28MTUT.29.2C_for_serverless_thin_clients_and_self-

[5] In addition to [6] that mentioned the idea of "a separate dust tier" in their future work, another paper[48] suggested moving parts of the UTXOS into secondary storage; however, we choose not to discuss it in detail as it assumes a separate UTXO for fee in every transaction[49], which is not true for Bitcoin unless a confidential transaction scheme is applied.

verifiable_prunned_blockchain
8. Todd P. 2016, "Making UTXO set growth irrelevant with low-latency delayed TXO commitments". bitcoin-dev mailing list.https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2016-May/012715.html
9. Bolton Bailey and Suryanarayana Sankagiri, "Merkle Trees Optimized for Stateless Clients in Bitcoin", ifca, 5th Workshop on Trusted Smart Contracts, WTSC21
10. Dryja T.,"Utreexo: A dynamic hash-based accumulator optimized for the bitcoinutxo set.", IACR Cryptol, ePrint https://eprint.iacr.org/2019/611.
11. https://bitcointalk.org/index.php?topic=21995.0
12. https://bitcointalk.org/index.php?topic=88208.0
13. Miller, A., Hicks, M., Katz, J., Shi, E.: "Authenticated data structures, generically", ACM SIGPLAN Notices 49(1), 411423 (2014)
14. Dan Boneh, Benedikt Bünz, and Ben Fisch, "Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains", Cryptology ePrint Archive: Report 2018/1188. https://eprint.iacr.org/2018/1188, last revised 20 May 2021.
15. Rasmus Dahlberg, Tobias Pulls, and Roel Peeters2, 'Efficient Sparse Merkle Trees Caching Strategies and Secure Non-Membership Proofs", NordSec 2016 Computer Science, DOI:10.1007/978-3-319-47560-8_13, Corpus ID: 3812588, https://eprint.iacr.org/2016/683.pdf
16. https://github.com/opentimestamps/opentimestamps-server/blob/master/doc/merkle-mountain-range.md, last visited 14/10/2021
17. https://crates.io/crates/merklemountainrange, last visited 14/10/2021
18. Colin Harper, https://www.coindesk.com/human-rights-foundation-gives-new-bitcoin-development-grants, 31May2021
19. https://youtu.be/HEKtDILPeaI, Bolton Bailey, "Merkle Trees Optimized for Stateless Clients in Bitcoin", Mar,2021
20. https://dci.mit.edu/utreexo, https://github.com/mit-dci/utreexo
21. Sergi Delgado-Segura, Cristina P´erez-Sol`a, Guillermo Navarro-Arribas, Jordi Herrera-Joancomart´ı, "Analysis of the Bitcoin UTXO set", In The 5th Workshop on Bitcoin and Blockchain Research.https://eprint.iacr.org/2017/1095.pdf
22. Robert Konrad & Stephen Pinto, " Bitcoin UTXO Lifespan Prediction", Stanford University, machine learning report & poster, Dec 2015.
23. https://bitcoin.org/en/release/v0.15.0#changes-to-internal-logic-and-wallet-behavior
24. Liu, Yulin; Zhang, Luyao; Zhao,Yinhong, "Replication Data for: "Deciphering Bitcoin Blockchain Data by Cohort Analysis", https://doi.org/10.7910/DVN/XSZQWP, Harvard Dataverse, V2, Mar 2021.
25. Erhardt M., "Simulation-based evaluation of coin selection strategies in Scaling Bitcoin", Milan, Italy, 8–9 October 2016
26. Anil Gaihre, Yan Luo, Hang Liu, "Do Bitcoin Users Really Care About Anonymity?An Analysis of the Bitcoin Transaction Graph" ,2018.
27. Jake Frankenfield, "Proof of Burn Cryptocurrency", Feb 2020, https://www.investopedia.com/terms/p/proof-burn-cryptocurrency.asp
28. Kostis Karantias, Aggelos kiayias, Dionysis Zindros,"Proof Of Burn, destroying money for fun and profit", 24th International Conference on Financial Cryptography and Data Security FC2020, Lecture Notes in Computer Science, Springer Vol12059, ISSN (Print)0302-9743,ISSN (Electronic)1611-3349, https://fc20.ifca.ai/
29. https://blockchair.com/bitcoin/address/1CounterpartyXXXXXXXXXXXXXXXUWLpVr; accessed 31/8/2021
30. https://blockchair.com/bitcoin/address/1111111111111111111114oLvT2; accessed 31/8/2021
31. https://gist.github.com/dcod3d/90fe6209a0b3ae815a6eaa2aef53524c

32. https://counterpartytalk.org/t/public-passphrase-sweepable-multisig/1535
33. https://www.news.com.au/finance/money/investing/what-is-dusting-how-hackers-are-exposing-holes-in-cryptocurrency-privacy/news-story/5a74d5d3a3e677e8111ab0e91927a894
34. https://bitcoin.stackexchange.com/questions/100437/bitcoin-cli-0-19-1-wallet-not-sending-from-addresses-with-closest-amount; accessed 4/9/2021
35. https://github.com/bitcoin/bitcoin/issues/20870; accessed 4/9/2021
36. https://txstats.com/dashboard/db/non-standard-outputs-statistics?; accessed 31/8/2021
37. Alin Tomescu, " Catena Efficient Non-equivocation via Bitcoin", MIT MAS.S62, Cryptocurrency Engineering and Design, lec7, https://youtu.be/CCeq5PChvuk, spring2018.
38. Ivan Mercanti, Stefano Bistarelli, Francesco Santini, "An Analysis of Non-standard Bitcoin Transactions", June 2018, DOI:10.1109/CVCBT.2018.00016, 2018 Crypto Valley Conference on Blockchain Technology (CVCBT)
39. https://bitcoin.stackexchange.com/questions/76541/whats-the-use-of-not-relaying-non-standard-transactions-if-anyone-can-still-use
40. Ivan Mercanti,Stefano Bistarelli, Francesco Santini, "Non-Financial Applications of Blockchains: Systematizing the Knowledge", https://www.frontiersin.org/articles/10.3389/fbloc.2019.00007/full
41. Konstantinos Sgantzos, "Implementing A Church–Turing–Deutsch Principle Machine on a Blockchain ", Department of Computer Science and Biomedical Informatics, University of Thessaly, Lamia, Greece, 17-07-2017
42. https://www.coindesk.com/bitcoin-utxo-98-percent-profit
43. Dhruv Bansal, "Bitcoin Data Science (Pt. 1): HODL Waves", April 2018, https://unchained-capital.com/blog/hodl-waves-1/
44. Rafael Schultze-Kraft, "Breaking up On–Chain Metrics for Short and Long Term Investors", Mar 2020, https://insights.glassnode.com/sth-lth-sopr-mvrv/
45. Rafael Schultze-Kraft, "No, Bitcoin Ownership is not Highly Concentrated – But Whales are Accumulating", Feb 2021 https://insights.glassnode.com/bitcoin-supply-distribution/
46. Glassnode,"Weekon-chain#35", https://mobile.twitter.com/glassnode/status/1432254092252835845
47. https://bitinfocharts.com/top-100-richest-bitcoin-addresses.html, last accessed 4/12/2021
48. Jingyu Zhang, Siqi Zhong, Jin Wang, Xiaofeng Yu, and Osama Alfarraj,"A Storage Optimization Scheme for Blockchain Transaction Databases", Computer Systems Science & Engineering, DOI:10.32604/csse.2021.014530, CSSE, 2021, vol.36, no.3.
49. https://bitcointalk.org/index.php?topic=5357803.0

Appendix A

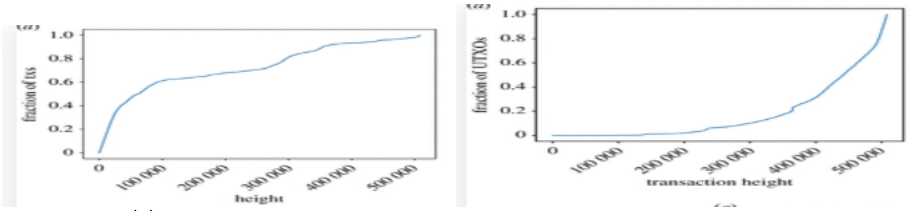
UTXO kinds

Various public sites presents running statistics and curves about the UTXO set; [24] compose the UTXOS according to the signing script, *blockchain.com* shows the continuous curve of the current number of UTXOS, *blockchair.com* and *btc.com* lets you observe and trace certain UTXO or address in detail (creation block, spending block,...etc). An interesting analysis and Simulation of the impact of coin selection strategy, wallet dependant, on the size and composition of the UTXO set (including dust ratio) is presented in [25];

however, most current wallets let the user chooses which UTXOS to spend in each transaction. In this appendix, we discuss the categorization of UTXOS according to the transaction or the use purpose type and its effect on their spending pattern. We introduce four types; coinbase UTXOS, burned UTXOS, dust UTXOS, and Non-standard UTXOS.

Coinbase UTXOS

As mentioned in section3, 101 is only the minimum lifespan for coinbase UTXOS, most samples taken had a much more; a specific incident of 167,300 block lifespan was found, rewards from blocks 2000,5000,9000,10000,20000 was still unspent till block 650,000. The usual pattern found was for miners to hold their rewards for some time, then accumulate them all (20-50 UTXOS) into one UTXO; a transaction that [26] called a *miner-merge* TX. Our sampling results that is not a complete enumeration of all values, was asserted by the following reverse curves presented in [21] enumerating till 500,000 blocks.



(a) coinbase UTXOS **(b)** total UTXOS set

Fig. the ratio of UTXOS created before a certain block

Therefore, we start from here introducing our first suggestion of storing coinbase UTXOS hashes in a separate Merkle Tree, since they have longer and completely opposite spending patterns than the fast ones. By replacing all living coinbase UTXOS with only one accumulated root value that changes only once per block without affect or being affected by the creation or spending of other UTXOS, a direct improvement is guaranteed. Proof sizes of coinbase UTXOs will be much smaller as their subtree size will never exceed the number of blocks ; when they got spent or created the change will be limited to their much smaller subtree, with only the root value change reported. On the other hand, their longer lives will not burden the proofs of other continuously changing kinds, and will only be summarized into one fixed root value for all fetched witnesses in a certain block.

Burned UTXOS

The concept of burning and proof of Burn, although existed from the early days of Bitcoin, has spread more widely in the last few years[27]. While burners are advised to use the OP_Return to not burden the UTXOS set with their burned coins that will never be spent, sometimes they do not; whether for hiding their aim till the burning take effect [28] or for any other reason as reality shows, that even when burning is public a considerable amount of users did not use OP_Return. These burned UTXOS that are going to live in the system state and consequently UTXOS Merkle Tree forever; sites [29,30]

show two publicly advertised burning addresses that contain 160,386 and 2843 UTXOS at the time of this writing. Hence, we suggest storing the hashes of burned UTXOS in a completely separate Merkle Tree that's expected to be append only and none of its nodes, except the root, will need to be fetched as a part of any proof. The recognition module of a burned UTXO could be more refined with further study&analysis of burning research and use cases, even just excluding the publicly known addresses will introduce an improvement.
The exact numbers of today are

(160,383+2,843)/(78.009*10^6)=0.163226/78.009 **~ 0.2092%** of the total number of current UTXOS burden can be removed from just two publicly advertised burning addresses that we knew; probably there are more.

Dust UTXOS

The term dust refers to currency values that are less than the minimum transaction fees needed to use them, a problem that exists in almost all cryptocurrencies, with more impact on UTXO-based ones due their nature that make it harder to control the remaining change value. Also Dust UTXOS are directly affected by the coin selection mechanism [25]; a selection strategy that seeks the closest match to the desired payment, like that of Bitcoin Core, could result in smaller change and consequently more dust values. The authors in [6] discussed the Bitcoin dust problem in detail, till block 500,000 when the size of the UTXO set was 60,216616 they measured 1.4% (842,892) holding exactly 1 Satoshi, 4%(2,408,264) holding less than 10^3 Satoshi, and 1.9%(1,144,115) holding 10^3-10^4 Satoshi. Probably most of those UTXOS were generated before the dust limit (546 Sat) was set by the Bitcoin Core, so in practice currently any remaining dust is added to the transaction fee. Dust UTXOS are a perfect candidate for a partition of expected size **~3.5m ~4.448%** of the UTXO set , since we do not expect any change in their considerable numbers, especially those holding less than the dust limit. Even if if those larger than the limit will ever get spent it will be through special transactions that will sweep a bunch of them together [31,32]; reserving locality of reference within their smaller subtree and removing an unnecessary burden from the main tree. Also, we can see from [33,34] that the very few exceptions of wallets that still force their coin selection strategy do not contradict; users may prefer to keep their dust and pay with a larger UTXO to save fee, and even when the wallet strategy was applied it combined many dust UTXOS together.
We recently found another estimate form [47]:
-Comparing the Bitcoin address distribution numbers to the number of addresses richer than 1$, we can calculate the no of addresses holding ≤1$
=(20,349,671+9,919,445+6,042,307+2,470,218+663,569+132,264+13,991+2062+86+3)-34,621,759
=4,962,947
-This means more than 4.9m Bitcoin addresses are holding less than 1$, and we know that an address can hold more than 1 UTXO; ie, this is a lowerbound Number of UTXOS (value less than 1$) ≥4,962,947 **~6.362%** of the UTXOS set
-It is true that the Bitcoin Core dust limit is 546 Sat ~0.3$, but a UTXO with a

value≤1$ is unlikely to be spent anyway (Probability approaching zero), add to this that the real number of UTXOS could be double or even triple this number. Hence, we can assume with a clear conscience that dust UTXOS are ~6.36%

Lastly, it's also worth mentioning that sending dust valued UTXOS to users has been reported as a possible attack to compromise their wallets[35], so the classifying step could serve also as a cautionary step against such attacks by marking the dust sending address as suspecious.

Non-Standard UTXOS

The second curve in [36] tell us that in Nov 29th, there are 422,822 non standard UTXOS **~ 0.542%** out of the complete UTXOS set. Non-standard UTXOS are the result of transactions that uses the Bitcoin ledger to securely transfer message data by writing it in place of the output public key; the value of the TXO is sacrificed as no private key could match the data written in the public key place. Those sacrificed UTXO will live forever, the problem of their burden was discussed 2016 in Bitcoin Developers mailing list[8]; MIT Catena lecture 2018 [37] is a perfect source to understand all about non-standard UTXOS; example companies are ***KeyBase*** and ***Blockstack***, the ***Catena*** project transfers just one UTXO back and forth to their publicly known address, no increase in the UTXOS number, and uses OP_Return to write their data We can see from the dates, the companies, and the curve slope&values that it is an old decaying concept; [38,39,40] describe in detail different kinds of non standard UTXOS; uses include time stamping, and anti replay key rotation Peggy back uses, bug fixing, and *sometimes an attack to burden the UTXOS set*. Since there is already a Bitcoin core ***IsStandard*** function, and like dust there is already also known threat of an attack usage, *we suggest storing non-standard UTXOS in a separate tree to enhance the performance and it may also serve as a cautionary step from the described attack*.

Appindex B

Pseudocode

In this appendix, we summarize the suggested classifying module to be added to the code of creating the UTXOS Merkle and fetching the needed proof. We introduce the partitioning idea in the most abstract sense; the suggested partitions could be modified or merged by any further study without changing the main concept.

The idea could be viewed as taking the Utreexo forest in a second dimension where trees represent the UTXO kind (2D forest); if the partitioning was done according to BTC value, every kind could be represented by a forest too as each group will reserve the same fast spending heuristic and will still be ***O(N)***. However, *with the smaller partitions* presented in section4 we do not find that necessary; the Utreexo forest used a maximum of ***logN*** roots, only one of them change with each insertion or deletion, to bound the worstcase proof length to ***logN***; we only add *a constant number of subroots*.

The implementation could be simple enough; another dimension added to Utreexo array of roots and a simple sequence of comparisons, if statements,

to handle the UTXO into the corresponding tree. The following Fig. a,b shows a simplified pseudo code for the partitioning described in section4, where we just have a 1D array, vector of roots, assuming the rest or default UTXO Merkle is represented by one root value whatever its structure is. A Similar one could be written if we partition according to BTC value, and also for any accumulator design other than the Utreexo forest.

```
//determine TX& UTXOs kind for all the UTXOS  j of current TX
    For All j
    {
        Kind=Analyze(TX.out[j]);
        Ins(TX.out[j], Tree[kind]);
    /*put the UTXOs in the appropriate tree*/
    Tree[kind]. modified=TRUE;
      /*this is to update this tree root in any cached proofs*/
    }
     For all inputs i
     Del(TX.inp[i]);
   /*the delete depends on creation kind, the UTXO is supposed to be linked to its appropriate tree, so when deletion the kind is known without checking*/
    //The delete should adjust the flag Tree[kind]. modified=TRUE;
```

Fig. a Pseudocode of steps added in maincode to handle partitions

```
int Analyze(struct UTXO)
{.    //Analyzing the Transaction kind
      If (UTXO== coinbase) return 1;
      If( IsStandard (UTXO) == FALSE) return 2;
       /*we favored non-standard to dust incase of overlap
         a decesion that could be easily modified*/
      If (UTXO.value ≤ Dust_lowerbound) {
            Suspected_adresses+= UTXO.address;    return 3; }
      If (UTXO.address IN BurnSet) return 4;
      return 0;     //The default is assumed kind 0
}
```

Fig. b Pseudocode of the partitioning module

Appindex C

Coinbase UTXOS sample data

A diagram demonstrating how the following values were found

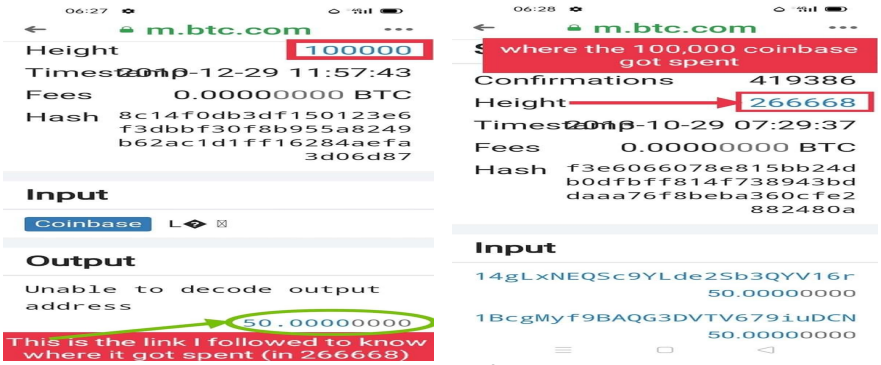


Fig. the address link with the target UTXO is followed to the transaction where it was spent, if the address is in many transactions we trace the UTXO by value and double check it is the one by pressing the link and getting back to the creating block (a)

Sample Data

Following the BTC value in the holding address
-300,000 got spent in 300,588
https://m.btc.com/1577232e5d734b6c9b4ddb5a35cef278611525d9ffd7c725dd44a93004b0543e
200,000 in 201,417
Life⟩1,417
https://m.btc.com/5c76eb4dfb0941856a229833ef05b2f5c669dadc98ed2a34ea11974cacba9dc7
-500,000 in 500,384⟩384
https://m.btc.com/2157b554dcfda405233906e461ee593875ae4b1b97615872db6a25130ecc1dd6
https://m.btc.com/fb999bf407393528ddaf640d0236ba1ea65f9b1fd979f5e63ea7bef6edff38ef
-600,000 in 600,662⟩ 662
https://m.btc.com/93955d40d918d014903843d258aada5c720a5d37afac7889268f459a97b148a3
https://m.btc.com/9807ac863d386928d705a21423e0d4801c7eaae5f6284977ca187957fe9b305c
-650,000 got spent in 650,154⟩154
https://m.btc.com/8143b3b341f665b22adcb8489158356c03f7c93cf4e4fa673d8518fa0fed95e4
https://m.btc.com/b663e948a1800f8be4e585c4654cb52cf364ddd8eceef0d58aa86be505c3170e

1-8,10,100, 200, 300,..,1000, 2000,5000, 9000, 10000, 20000, 30000, 40000 all Unspent
⟩lifetime≥ 650,000?!

9⟩161, 8000⟩146630
(a lot, about 80, in the same range from blocks 8000 to max I think 32,000+ ie all lifespan≥113000

https://m.btc.com/29483c3b6775025a9ee618dfb3d586b3bb8961ec3911dcc5944748e932f15723)
50000⟩4778
(with a lot in the same TX & thus range)
60000⟩Unspent, 70000⟩331, 80000⟩Unspent, 90000⟩121, 99368⟩167300
100000⟩166668, 110701⟩155967
There r 17 UTXOs with lifespan exactly bet the last three
(All got spent in the same transaction
https://m.btc.com/f3e6066078e815bb24db0dfbff814f738943bddaaa76f8beba360cfe2882480a)

150000⟩101, 200000⟩1417, 300000⟩588, 350000⟩171, 400000⟩101, 450000⟩176, 500000⟩384, 549082⟩132, 600000⟩662, 650000⟩154, 650001⟩13399, 660000⟩143, 665000⟩118, 669668⟩106, 670000⟩209
675000⟩548, 676779⟩33139, 677065⟩11302, 680000⟩923, 681000⟩Unspent
681188⟩17704, 681500⟩5011
Following the address of 681500 , also 676741 contains about 40 coinbase UTXOs (on 31/5/2021)
https://m.btc.com/1QEiAhdHdMhBgVbDM7zUXWGkNhgEEJ6uLd
Then they all got spent on block 688367, ie some lifespan=20,626
https://m.btc.com/2a44f5c3f553940750b1cc6b95e9a2361dabdc16bf1cb7ad410b6bb4b41bff52

Then we sampled more from the recent blocks as we thought the spending behavior may have changed through the years, especially comparing with the early days of Bitcoin say before block height 150,000.
681807⟩191, 681975⟩16260, 682000⟩178, 682324⟩1041, 683000⟩215
683208⟩2322, 683529⟩2001, 683813⟩26105, 684000⟩163, 684028⟩1502
684662⟩25256, 684811⟩719, 684880⟩650, 684906⟩624, 685000⟩105
685052⟩478, 685140⟩13095, 685245⟩285, 685402⟩128, 689602⟩1317
690619⟩19299, 705093⟩4825

650001⟩13399
Here, block 663,400 (28/12/2020) is like a sink TX with about more than 100 input to 1 output
https://m.btc.com/60916b6158148f9b1c95ff2024528c56ea17ad8365ac183214c7753b6c3857bf
-Not all the inputs are coinbase UTXOs, some coming from previous regular Transactions, for example the first input was created in block 623421
(lived even longer 663400-623421=40,000-21
=39,979)
https://m.btc.com/5ac9cc70158e8126bb215c6e491a8dafa1530ba346f4b864ba9261ef68611e2f

Plots of the Sampled Data (Coinbase UTXOS Lifespan)

Here we tried to draw different ranges of block heights as the last range contain more sampled data, and we also added a logarithmic scale plot to

present a more clearer view since age values varies from 167,000 to 101
Notes:
-The value seems approaching zero is 101, and the part that resembles a straight line is in the range with less number of samples compared to the oscillating part that constitutes 67% of the samples.
-We did not include the earlier samples that are not spent till now (1-8,10,100, 200, 300,..,1000, 2000,5000, 9000, 10000, 20000, 30000, 40000) since they will be having age≥650,000 that will make the rest of values look as they are approaching zero.