Dietcoin: Hardening Bitcoin Transaction Verification Process For Mobile Devices

Davide Frey
Univ Rennes, Inria, CNRS, IRISA, France
davide.frey@inria.fr

Marc X. Makkes
Vrije Universiteit, The Netherlands
m.x.makkes@vu.nl

Pierre-Louis Roman
Univ Rennes, Inria, CNRS, IRISA, France
pierre-louis.roman@irisa.fr

François Taïani
Univ Rennes, Inria, CNRS, IRISA, France
francois.taiani@irisa.fr

Spyros Voulgaris
Athens University of Economics and Business, Greece
voulgaris@aueb.gr

ABSTRACT

Distributed ledgers are among the most replicated data repositories in the world. They offer data consistency, immutability, and auditability, based on the assumption that each participating node locally verifies their entire content. Although their content, currently extending up to a few hundred gigabytes, can be accommodated by dedicated commodity hard disks, downloading it, processing it, and storing it in general-purpose desktop and laptop computers can prove largely impractical. Even worse, this becomes a prohibitive restriction for smartphones, mobile devices, and resource-constrained IoT devices.

In this demo, we present an implementation of Dietcoin, a Bitcoin protocol extension that allows nodes to perform secure local verification of Bitcoin transactions with small bandwidth and storage requirements. This demo presents and benchmarks the main features of Dietcoin that are important for today's cryptocurrencies and smart contract systems, but are missing in the current state-of-the-art: (i) allowing resource-constrained devices to verify the correctness of selected blocks locally without having to download the complete ledger; (ii) enabling devices to join a blockchain quickly yet securely, dropping bootstrap time from days down to a matter of seconds; (iii) providing a generic solution that can be applied to other distributed ledgers secured with Proof-of-Work.

PVLDB Reference Format:
Davide Frey, Marc X. Makkes, Pierre-Louis Roman, François Taïani, Spyros Voulgaris. Dietcoin: Hardening Bitcoin Transaction Verification Process for Mobile Devices. *PVLDB*, 12(12): 1946-1949, 2019.
DOI: https://doi.org/10.14778/3352063.3352106

Proceedings of the VLDB Endowment, Vol. 12, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3352063.3352106

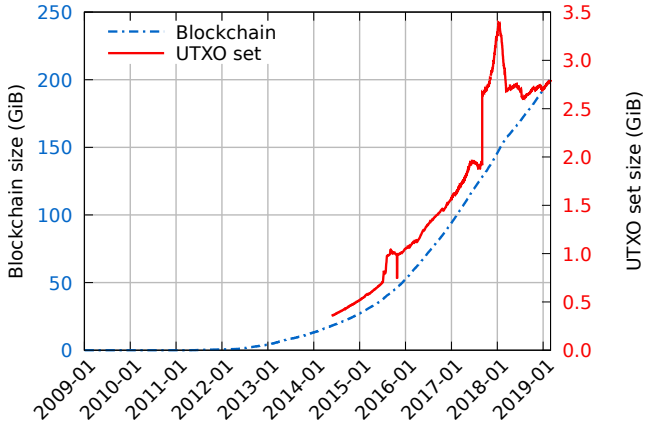

Figure 1: The size of both the Bitcoin blockchain and of its UTXO set have increased sixfold over the past four years.

1. INTRODUCTION

Within a decade, blockchains have become extremely popular, and have been used to implement several widely-used cryptocurrencies [7] and smart-contract services [3, 8]. A blockchain implements a *tamper-proof distributed ledger* in which transactions can be recorded in an irrevocable, or probabilistically irrevocable, manner. Recorded transactions are stored in *blocks*, which are incrementally *chained* in order to form an append-only list that is synchronized between participants via peer-to-peer exchanges. The security of these chaining mechanisms leverages cryptographic tools and Byzantine agreement mechanisms. Bitcoin, for instance, prevents spam blocks by using computationally costly Proof-of-Work and pairs it with a rule prioritizing the chain with the most aggregated work to ensure convergence on the state of the blockchain. This combination makes it practically infeasible for individual participants to revoke past transactions due to the inconceivable computational costs involved, while it remains possible for any participant to verify the validity of a blockchain's entire history.

Although this verification is possible for any participant, the associated costs are far from trivial. The verifying node must (i) download the entire blockchain and (ii) verify every

single block. Downloading an entire blockchain is feasible yet largely impractical, in particular for popular blockchains whose total history size can lie in the order of hundreds of gigabytes. Downloading and processing data of this magnitude is way beyond the capabilities of most resource-limited devices, such as smartphones and IoT devices, which is a major barrier to adoption since these devices are likely to serve as digital wallets for micro-transactions due to their handiness and omnipresence.

The Bitcoin blockchain, for instance, has grown to 200 GiB as of March 2019 (Figure 1), and historically follows an exponential growth despite having a capped block size (which is the subject of vigorous debates). As blockchains are ever-growing structures, the problem can only become more acute. In addition to downloading the blockchain, the verifying node must also check its consistency block by block, a linear process that can take hours on high-end machines and much longer on smartphones.

Assuming average mobile connection bandwidths [1], it takes 30 hours to download the necessary 200 GiB of blocks at 15.1 Mib/s on 4G, and even up to 74 hours at 6.1 Mib/s on 3G. Moreover, verifying the blockchain typically takes longer than downloading it, particularly on computation-constrained devices designed for optimal battery longevity.

The exorbitant price of a full chain verification makes it unrealistic for low-resource devices to implement a blockchain protocol fully. Some blockchain systems, such as Bitcoin, therefore enable nodes to perform varying degrees of verification: *full nodes* verify everything, while light nodes, known as *Secure Payment Verification nodes* (*SPV nodes*) only verify a small fraction of the data (i.e., the block headers).

To protect themselves against counterfeited transactions, full nodes keep track of unspent funds in a structure known as the set of *Unspent TransaCTion Outputs* (*UTXO set*). The UTXO set (i.e., the state of the ledger) is aggregated by linearly parsing the entire blockchain (i.e., the list of updates), making it expensive to construct, to exchange (since it weights 2.8 GiB as of March 2019, see Figure 1), and to maintain. This accumulation of costs prevents low-resource devices from relying on the UTXO set. However, since these devices can be used as Bitcoin (mobile) payment terminals, especially in regions with no reliable card-based payment infrastructures, their impaired security, in part due to the current limitations of the Bitcoin protocol, could be significantly improved by harnessing the UTXO set.

In this demo, we show an implementation of the *Dietcoin protocol* [4, 5] with its associated *diet nodes* that bridge the security gap between full nodes and SPV nodes. The demonstration contains two incremental scenarios to show its efficiency and effectiveness. Scenario 1 shows the initialization phase of a diet node running on a phone. This phase only takes seconds and shows the associated *energy*, *bandwidth*, and *verification cost* of running a diet node on the phone. Scenario 2 shows the diet node perform a transaction verification including showing the aforementioned costs. For both scenarios, a visitor merely needs to either click the "restart" button, to re-initialize and download the blockchain from scratch, or click the "verify transaction" button, to execute a transaction and perform verification. Finally the demo is accompanied by a poster to elaborate on how the verification mechanism works.

The remainder of this paper is organized as follows. Section 2 describes the efficient verification of transactions in the Dietcoin system, Section 3 describes the implementation of Dietcoin, Section 4 discusses the demo scenarios, and finally, Section 5 concludes.

2. DIETCOIN OVERVIEW

Dietcoin aims at increasing the security of low-resource devices by offering diet nodes that are able to verify the correctness of blocks, as well as subchains of blocks, at a reasonable cost. In short, diet nodes perform secure queries to full nodes for their local replica of the UTXO set and limit their queries to what is required to verify specific blocks.

To that extent, Dietcoin makes three key modifications to Bitcoin:

1. the UTXO set is split into *UTXO shards* to ensure affordable bandwidth consumption for diet nodes (Section 2.1);

2. the integrity of the UTXO shards is preserved (i) by building a Merkle tree of the shards and (ii) by storing its root, which we call the *UTXO Merkle root*, into blocks, to protect its integrity in the way transactions are protected (Section 2.2);

3. the authoritative UTXO Merkle root stored in each block allows diet nodes to fully verify individual blocks as well as subchains of blocks (Section 2.3).

We end this overview by providing in Section 2.4 a few usages enabled by these modifications made to Bitcoin.

2.1 Sharding the UTXO set

Full nodes store the UTXOs resulting from each block's transactions into UTXO shards. As the UTXO set evolves with each block, recently modified UTXO shards are stored by full nodes and are available for query by diet nodes who wish to fully verify recent blocks. To maintain small shards, their size is capped such that the average of all shard sizes must not cross a predefined limit $\mathcal{M}$ (e.g., 1 KiB).

The sharding policy uses the first k bits of each UTXO to determine the UTXO shard it belongs to. Since the first 32 bytes of a UTXO is the SHA-256 transaction hash it references, using the first k bits for indexing resembles random indexing. This particular indexing results in shards of theoretical homogeneous sizes. Each full node locally increments k when the cap $\mathcal{M}$ is crossed, which doubles the number of shards and halves the average shard size in the process.

2.2 Committing the UTXO set into blocks

In addition to UTXO shards, full nodes also maintain a Merkle tree to enable diet nodes to verify the integrity of said shards. The root of that tree is committed into blocks and is subject to verification, hence is rejectable, by each validating node. A miner committing an erroneous UTXO Merkle root is at risk of seeing his block rejected and consequently losing the rewards that come with it.

Since UTXOs are indexed by their first k bits, there are 2^k shards at all times which results in a complete UTXO Merkle tree with 2^k leaves all at the same height. Because only a subset of all UTXO shards are modified when a block is added to the chain, only a subset of the UTXO Merkle tree leaves are modified, while the vast majority of the tree remains unchanged. Using shards makes the UTXO Merkle

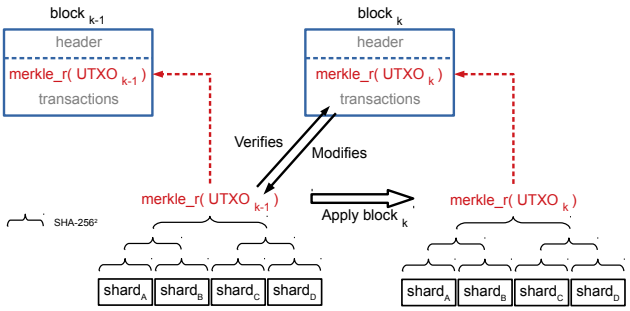

Figure 2: The UTXO set is updated at every block validation. For a counterfeited block to be validated by diet nodes, a malicious miner has to forge at least two consecutive blocks: the first block B_{k-1} containing a fake UTXO Merkle root of UTXO_{k-1}, and the second block B_k spending fake coins validated by the fake UTXO_{k-1}.

tree updatable in-place and enables diet nodes to keep unchanged sections of the tree in a cache to further reduce their bandwidth consumption.

To ensure a smooth adoption, Dietcoin must fit within the existing Bitcoin consensus rules. The commitment is therefore stored in a transaction output that Bitcoin nodes ignore. Although the Dietcoin protocol is compatible with the Bitcoin protocol, the UTXO set is stored differently. Diet nodes must then connect to full nodes running Dietcoin to obtain the needed UTXO shards.

2.3 Enabling block and subchain verification

Thanks to sharding the UTXO set and committing its Merkle root into blocks, diet nodes are able to securely query UTXO shards to verify the correctness of blocks' transactions. However, diet nodes must also be able to verify the correctness of the now committed UTXO Merkle root.

As depicted in Figure 2, a diet node can verify the UTXO Merkle root of block B_k by acquiring (i) the UTXO Merkle root stored in block B_{k-1}, (ii) all the UTXO shards modified in B_k and (iii) the partial UTXO Merkle tree needed to ensure the integrity of all the downloaded shards. Once the integrity of each UTXO shard has been verified, the diet node updates its copies of the UTXO shards by removing from them all the UTXOs spent in B_k, and by adding all the UTXOs created in B_k. This operation is similar to the one performed by full nodes, the only difference is that diet nodes do not have to possess the shards that are not needed to verify the transactions in B_k. After updating the shards, the diet node updates, in turn, their corresponding leaves in the UTXO Merkle tree as well as all their parent nodes in the tree. The root of the now updated Merkle tree should be equal to the value committed in B_k, provided the block is correct, which completes the full verification of block B_k including both its transactions and UTXO Merkle root.

This block verifying process can be iterated towards the next blocks, by relying on the trust that the diet node has in the verified UTXO Merkle root in block B_k. Dietcoin, therefore, can be used to verify both individual blocks and subchains of blocks. Assuming a subchain of ℓ blocks (e.g., 6 blocks[1]), a diet node can increase its trust in B_k by verifying the correctness of the subchain $(B_{k-\ell+1}, ..., B_k)$ to ensure that none of these blocks have been counterfeited. While a block is already costly to create for an attacker, a subchain of blocks is exponentially more costly to counterfeit [6]; verifying that they are all correct can exponentially increase the trust a diet node has in B_k therefore drawing its trust in B_k much closer to that of a full node.

2.4 Dietcoin usages

The main usage enabled by Dietcoin, which is the one presented in this demo, is to enable low-resource devices to verify the correctness of individual blocks and subchains of blocks.

As second usage, Dietcoin enables low-resource devices to check the inclusion of specific UTXOs into the UTXO set, even if these UTXOs were created in old transactions. Diet nodes can infer from a UTXO the shard it should belong to, and query another node for that specific shard at any point in time. This usage allows low-resource devices to ensure that (i) specific transactions have been committed into the blockchain, and that (ii) their UTXOs are still spendable and have not been stolen.

Finally, full nodes too can benefit from Dietcoin as they can quickly bootstrap into functioning verifying nodes thanks to the UTXO set commitment. Full nodes can skip the verification of most blocks, choose to download all the UTXO shards at a point in time and only verify the correctness of the blocks that are part of the, possibly long, head of the blockchain. Alternatively, full nodes can start their bootstrap by first downloading all the UTXO shards forming the most recent UTXO set, then parse the entire blockchain backwards. Doing so enables them to quickly verify new blocks, while verifying the chain in the background.

Both UTXO proof of inclusion and fast bootstrapping full node have been presented in other non peer-reviewed proposals made by the Bitcoin community [2], but these proposals however do not enable affordable subchain verification.

3. IMPLEMENTATION

The experimental setup for Dietcoin is designed to produce data that is as realistic as possible. To this end, Dietcoin (i) is deployed in a realistic environment, (ii) is based on a mature open-source code, and (iii) uses a large and real dataset. We chose to extend the de facto standard Bitcoin java library *bitcoinj*[2] with the features of Dietcoin to have diet nodes executable on Android smartphones. We use the Bitcoin blockchain as a large public dataset and we have the actual blocks replayed to the diet node.

The software architecture for the demonstration is separated into three components (Figure 3 (right)). A Bitcoin full node (using *bitcoin-core*[3]) serves as a verified store for the blockchain and as an interface to the outer network. A Dietcoin replayer is used as a middleman: it fetches blocks from the Bitcoin full node, enriches them with the UTXO Merkle root, locally stores the UTXO shards and UTXO Merkle tree, and interacts with the diet node. Finally, the diet node deployed and monitored on an Android smartphone makes all its queries to the Dietcoin replayer as it would to a regular Dietcoin full node.

[1] `https://bitcoin.org/en/payment-processing-guide#verifying-payment`

[2] `https://github.com/bitcoinj/bitcoinj`

[3] `https://bitcoin.org`

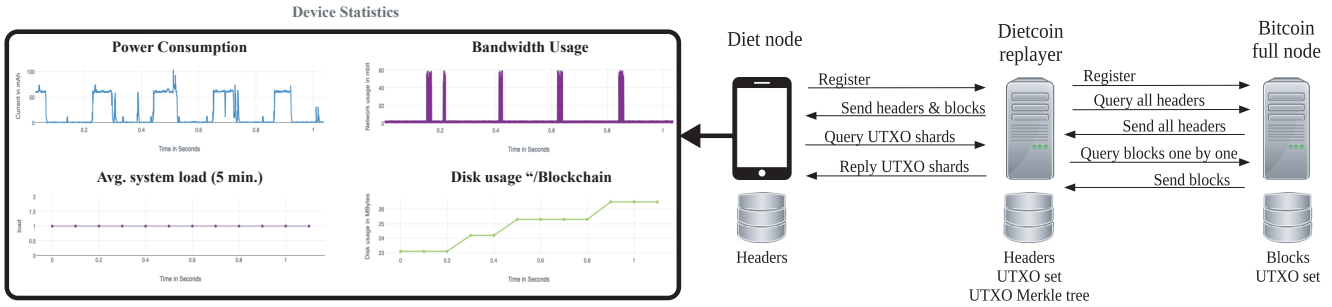

Figure 3: The demo showing real-time system statistics of a diet node performing a transaction verification within the Bitcoin blockchain replaying setup.

4. DEMONSTRATION

The demonstration of Dietcoin is performed on several devices which include a Raspberry PI 3+, the smartphones LG Nexus 5, LG Nexus 5X, and Huawei Nexus 6P accompanied by a Monsoon power monitor and a laptop that displays system statistics from these devices. The demonstration contains two scenarios, which are paired with a poster illustrating the workings of Dietcoin. The first scenario shows the difference in bootstrap time of a diet node compared to that of a Bitcoin full node and a Bitcoin SPV node. The second scenario shows the resource consumption impact of performing a transaction on the IoT device.

Demo GUI. Figure 3 (left) shows a screenshot of the demo GUI. The GUI is part of a device monitor that collects data from the demonstration resource-constrained device and displays it to the user. The statistics depicted on the GUI are system load over a 5 minutes window, current CPU load, current storage usage used for the blockchain, the power consumption of a device in mAh, and network usage in Mib/s. In addition, but not shown in Figure 3, are the accumulated network usage as well as the total processing time of each implementation. Users can select different devices and can compare different clients.

Demo Dataset. The demonstration uses the ever growing full Bitcoin blockchain which is 200 GiB big as of March 2019. The Bitcoin blockchain contains over 10 years of public data since its launch in January 2009 and is comprised of 567,000 blocks and 391,200,000 transactions.

Demo Scenarios. This demo provides two scenarios to showcase the main properties of Dietcoin.

Scenario 1: a bootstrapping comparison between a Bitcoin full node, a Bitcoin SPV node, and a diet node. In this scenario, users can initiate all types of clients, i.e., remove all previously downloaded data, and start from the genesis Bitcoin block. The statistics of the device is reset on startup, and is depicted as shown in Figure 3. Users can change the time frame of the statistics to get an overview of the involved computational, network, and system costs.

Scenario 2: transaction costs comparison between a full Bitcoin node, a Bitcoin SPV node, and a diet node. Users can press a button that, after bootstrapping, verifies a specific transaction which can be selected from a list. After pressing the "verify transaction" button, the system verifies that the selected transaction is valid, i.e., by requesting the UTXO Merkle tree from a replayer/full node. Users can directly see what the network, storage, and computational costs for verifying a transaction are.

5. CONCLUSION

In this demo, we demonstrate through Dietcoin the benefits of sharding the state of the distributed ledger of Bitcoin and making it available for other nodes to query. This demo allows the audience to securely interact with the Bitcoin cryptocurrency without having to suffer through the costly requirement of downloading the entire ledger.

Acknowledgments

This work has been partially funded by the Region of Brittany, France, by the Doctoral school of the University of Brittany Loire (UBL), by the French National Research Agency (ANR) projects *SocioPlug* under contract ANR-13-INFR-0003 and *OBrowser* under contract ANR-16-CE25-0005, by the Dutch SIDN Fonds contract 172027 and by the Dutch Organisation for Scientific Research (NWO) under contract 629.009.014.

6. REFERENCES

[1] Measuring mobile broadband performance in the UK: 4G and 3G network performance. Ofcom Report, 2014. `https://www.ofcom.org.uk/__data/assets/pdf_file/0014/32054/mbb-nov14.pdf`.

[2] B. Bryan. [bitcoin-dev] Protocol-Level Pruning, Nov. 2017. `https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2017-November/015297.html`.

[3] T. D. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen. Adding concurrency to smart contracts. In *PODC*, 2017.

[4] D. Frey, M. X. Makkes, P.-L. Roman, F. Taïani, and S. Voulgaris. Bringing secure Bitcoin transactions to your smartphone. In *ARM workshop*, pages 3:1–3:6. ACM, 2016.

[5] D. Frey, M. X. Makkes, P.-L. Roman, F. Taïani, and S. Voulgaris. Dietcoin: shortcutting the Bitcoin verification process for your smartphone. Mar. 2018.

[6] L. Kiffer, R. Rajaraman, and a. shelat. A Better Method to Analyze Blockchain Consistency. In *CCS*. ACM, 2018.

[7] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[8] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2014.