Empirical Analysis of On-chain Voting with Smart Contracts

Robert Muth and Florian Tschorsch

Technische Universität Berlin, Germany
{muth, florian.tschorsch}@tu-berlin.de

Abstract. Blockchains and smart contracts promise transparency, verifiability, and self-enforcing agreements. Against this background, novel use cases such as decentralized governance platforms that implement voting to collectively manage funds have emerged. While a number of arguments against blockchain-based voting exist, we still see a relevance. In this paper, we therefore present a quantitative analysis of the Ethereum blockchain with respect to voting. To this end, we develop a blockchain analysis toolchain that we use to analyze 3 173 smart contracts on the Ethereum Mainnet with voting functionality. We extract insights on the complexity of deployed voting methods and reveal a trend towards a centralization of funds, i.e., five smart contracts manage 98 % of funds comprising more than four million USD. We additionally analyze the feasibility of on-chain voting for Ethereum as well as other well-established blockchains that are used for voting, i.e., Bitcoin and Dash.

Keywords: Blockchain · Analysis · Voting · Smart Contract.

1 Introduction

The blockchain's integrity and transparent storage space make it tempting to implement blockchain-based online voting [8, 10, 14] as everyone can verify the correct execution. In particular, blockchains such as Ethereum [27], which provide an opportunity to implement smart contracts [24], inherently allow to verify whether a vote was stored and counted correctly. However, it has been shown and argued that blockchain-based online voting has fundamental issues [8, 19], including security [18, 23] and privacy [9] problems.

While blockchain-based online voting certainly polarizes, on-chain voting is still being used for reasons such as the decentralized governance of funds. Most prominently, decentralized autonomous organizations (DAOs), e.g., the DAO [11], allow fundraising and enable stakeholders to manage the distribution of funds with on-chain voting. Smart contracts render the decision-making process transparent and self enforcing. Since its debut in 2016, the DAO raised approximately 150 million USD, but at the same time lost about 60 million USD due to an exploit [1]. While we distance ourselves from the idea of blockchain-based online voting, e.g., for official elections, we argue that *on-chain voting* still requires attention and further research.

In this paper, we show the relevance of on-chain voting and derive limitations in terms of scalability and transaction costs. To this end, we scan the Ethereum Mainnet for smart contracts with voting functionality and analyze their usage with respect to registered votes, gas costs, and fundings. In order to understand the scalability potential of on-chain voting, we analyze past residual blockchain capacities of Ethereum and evaluate the feasibility of small and large-scale votings. We also look beyond Ethereum and discuss other leading blockchains, including Bitcoin [17] and the governance network of Dash [6]. We provide a publicly available repository with the collected data sets and our analysis pipeline. Our presented database driven analysis approach is compatible with Google BigQuery and therefore does not require any advanced setup.

In our empirical analysis, we found 3 173 deployed Ethereum smart contracts related to voting, which currently hold 11 794 ETH, or more than 4.5 million USD (as of October 30, 2020). From these smart contracts, we identified 88 instances of the DAO (deployed smart contracts that are based on the original DAO source code), which in total received 5 928 votes, so far. Over the past years, voting smart contracts in general accumulated and processed 29 337 ETH. Our analysis suggests a continuously high amount of monetary investments in and interaction with voting smart contracts, indicating a high popularity and relevance. Besides the relevance, we conclude that blockchain voting suffers from scalability issues that render large-scale votings either not feasible in a reasonable time, or very expensive, or both.

The main contributions of this paper can be summarized as follows:

- We develop an analysis pipeline to reveal voting smart contracts on the Ethereum blockchain and present an overview of key metrics, which emphasize the relevance of on-chain voting (see Section 3)
- We assess the limitations of on-chain voting with a model-based comparison of blockchain specifications as well as an analysis using historic block data (see Section 4)
- We give an outlook on other relevant blockchain with on-chain voting, i.e., Bitcoin and Dash (see Section 5)

In addition to our main contributions, we discuss related work in Section 2 and conclude the paper in Section 6.

2 Related Work

There is a large body of work on blockchain-based voting, proposing various designs to conduct votings using blockchain technologies [5, 10, 12, 16]. Most notably, McCorry et al. [16] developed a smart contract for boardroom voting with maximum voter privacy. Since we do not propose any new voting schemes, these contributions are orthogonal to our work.

In this paper, we analyze the multitude of on-chain voting regardless of any specific use case or property. A series of contributions investigate blockchain

data with respect to various other aspects, including privacy [2, 22], data storage [15], and smart contract metrics [20]. Moreover, model-based analysis on the security [13] and scalability [4] of blockchains in general exist. Specific to voting, Heiberg et al. [8] evaluate the trade-offs of blockchain-based voting on a qualitative level. They discuss aspects such as complexity, costs, and scale, which go in a similar direction as our paper. We complement their discussion however with an empirical analysis and reveal new insights, for example, on the magnitude of on-chain voting.

Methodically similar to our approach, are [7, 20, 21, 26]. Victor and Lüders [26] inspect the Ethereum blockchain for token implementations, which are managed by the ERC-20[1] smart contract template. While EIP-1202[2] proposes a similar standard for voting smart contracts, it is not as established as the ERC-20 compatible token standards. Fröwis et al. [7] search for token-related behavior with symbolic execution analysis techniques and compare the effectiveness of both methodologies. The diversity of voting schemes, features, and privacy mechanisms make it more difficult to identify voting smart contracts by their bytecode. We therefore propose an analysis pipeline that uses generic voting signatures from other sources in addition to established method signatures. In contrast to automated smart contract inspection, the authors of [20, 21] present approaches that are based on manually collected exchange listings and corresponding source code publications on CoinMarketCap and Etherscan.

3 Relevance of On-chain Voting

In this section, we reveal the magnitude of on-chain voting in Ethereum. We are particularly interested in the diversity of voting smart contracts with respect to cost and fundings.

3.1 Analysis Toolchain and Methodology

Typically, analyzing blockchains requires a synchronized node with all valid transactions. With Geth, the Ethereum foundation provides such a node, which has been optimized to save computational resources and memory. As it turns out, the very data-efficient data structures make it difficult to quickly analyze historic data. For this reason, we instead used Google BigQuery[3] as source to Ethereum Mainnet transactions. BigQuery is a Google Cloud service for big data analysis, which provides a public dataset with all current Ethereum transactions, block details, and smart contracts in a SQL database. As shown in Figure 1, we use BigQuery as transactions source and to execute complex SQL queries for analysis. The advantage of SQL databases is the ability to index past transactions and query them efficiently (at the cost of additional storage and memory consumption which BigQuery compensates with cloud resources). We

[1] `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md`
[2] `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1202.md`
[3] `https://cloud.google.com/bigquery`

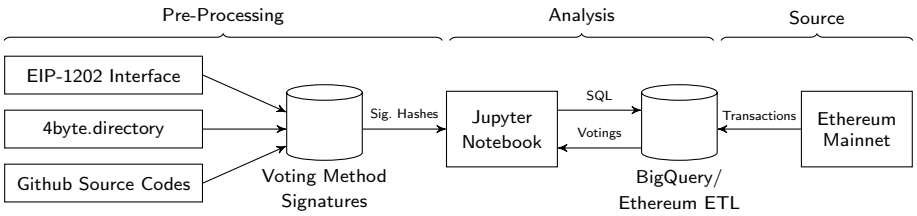

Fig. 1. Our blockchain-based voting analysis toolchain with a Jupyter Notebook and BigQuery (or Ethereum ETL) based on given pre-processed method signatures.

developed a Jupyter Notebook, which manages the analysis process, i.e., preparing input data from pre-processing, compiling SQL statements, monitoring the execution, and preparing the results. Alternatively to our cloud-based approach, the database can be generated locally using a full node and Ethereum ETL[4] without BigQuery.

While smart contracts are generally stored publicly on the Ethereum blockchain, only the compiled bytecode, i.e., EVM code, is available. Similar to high-level programming languages, the original source code compiles to an assembly-style language. To this end, compilers remove comments and substitute identifiers, which render the bytecode difficult to understand without the original source code. In addition, method signatures of smart contracts, i.e., method name and parameter list, are represented by a hash pointer. More specifically, the first for 4 bytes of a method signature's Keccak (SHA-3) hash value are used to point to the respective stack code position. Since Keccak is a cryptographic hash function, it is not possible to infer the method signature from the hash value directly. Hence, it is neither straight forward to search for a certain type of smart contract nor for a partial method signature.

In order to analyze the Ethereum blockchain, we searched for hash values of method signatures that are usually part of voting smart contracts. As shown in Figure 1 as part of the pre-processing, we collected the hashed method signatures of the EIP-1202 voting interface, which provides a standardized set of methods for voting. In addition, we used the Ethereum Function Signature Database[5], which provides a list of method signatures and their corresponding hash values based on known smart contract source codes and user submissions. We used the database's RESTful API to search for methods containing 'vote', 'voting', or 'ballot'. As a result, we get a list of method signature and hash value tuples, which are related to voting. We use these tuples to retrieve the smart contracts that actually implement the respective method. Finally, we analyzed the source code of the DAO smart contract on Github for identifying transactions to the original instance and deployed copies with them same interface methods.

[4] `https://github.com/blockchain-etl/ethereum-etl`
[5] `https://www.4byte.directory`

Inevitably, the approach may lead to some positives as well as false negatives. For example, generic method signatures lead to a false classification of some smart contracts, e.g., `setStatus(...)` of the EIP-1202 or `dropVotes(...)`. We also encountered hash collisions that indicated voting methods in a smart contract but did not belong to voting upon closer inspection. For example, the method signatures `voting_var(address,uint256,int128,int128)` and `totalSupply()` share the same hash value `0x18160ddd` and lead to false-positives. In an attempt of manual inspection, we excluded these instances for our analysis. In order to prioritize precision (over sensitivity), we considered smart contracts that implement at least two method signatures related to voting only. Since the bytecode in the blockchain remains a black box, though, we cannot exclude false classification entirely.

The described methodology enables analyses of Ethereum smart contracts in general and can be used to reveal a multitude statistics. We used it to analyze voting smart contracts with respect to scale and gas cost in general and the interaction with these contracts in particular. We inspected the Ethereum blockchain for the timespan between October 16, 2017 and October 30, 2020. Moreover, we developed a Jupyter Notebook[6] which connects to BigQuery, our own local data records (e.g., historical exchange rates), and other external data sources. A data dump of the following results, the implementation to gather the data set independently, and our full analysis pipeline to reproduce the results is publicly available on GitHub.[7]

3.2 Voting Complexity

In total, we found 1 458 relevant method signatures related to voting, which are implemented in 5 185 smart contracts. Overall, 1 272 059 transactions interacted with these smart contracts and called 129 855 times one of the voting methods. After data cleaning, 3 173 voting smart contracts remain and are subject of the following analysis.

In Table 1, we show the ten most often called voting methods and their average consumed gas. None of the deployed voting smart contracts implemented EIP-1202 completely, but 82 of them implemented at least a subset of its standardized method signatures. While most of the method signatures in Table 1 are not surprising, methods 5, 6, and 9 let us expect a *commit-and-reveal* voting scheme, where voters submit their vote cryptographically concealed, e.g., by using a hash function, and reveal their individual vote later with another transaction. Since such a scheme is more complex, it typically requires more gas.

Method signatures with more than one parameter mostly belong to smart contracts that conduct multiple votings and allow to specify a proposal. For example, most calls with method signature 7 belong to a DAO smart contract that conducts multiple votings, where the `byte32` parameter references the proposal and the `uint256` parameter encodes the user's choice.

[6] `https://colab.research.google.com/drive/1oIxMjJu7LQvSMnXiIgC9S_5CgGA_5d2R`
[7] `https://github.com/robmuth/blockchain-voting-analysis`

Table 1. Top ten voting methods with respect to their number of calls.

	Calls	Hash	Signature	⌀ Gas	⌀ Gas Price
1	80 676	`0x0121b93f`	`vote(uint256)`	71 k	2.4 Gwei
2	6 996	`0xb384abef`	`vote(uint256,uint256)`	31 k	28.3 Gwei
3	6 420	`0xfc36e15b`	`vote(string)`	32 k	3.2 Gwei
4	4 534	`0xddb6e116`	`vote(uint16)`	47 k	3.7 Gwei
5	2 930	`0x6cbf9c5e`	`commitVote(uint256,bytes32,..`	164 k	3.8 Gwei
6	2 624	`0x5e8254ea`	`commitVoteOnProposal(bytes32,..`	110 k	7.0 Gwei
7	2 161	`0x9ef1204c`	`vote(bytes32,uint256)`	151 k	9.6 Gwei
8	2 124	`0xcff9293a`	`vote(uint32,uint32)`	51 k	12.1 Gwei
9	2 009	`0xb11d8bb8`	`revealVote(uint256,uint256,..`	62 k	3.4 Gwei
10	1 817	`0x3850f804`	`castVote(uint256,uint256[],..`	139 k	41.1 Gwei

Table 2. Top four smart contracts with respect to their funds.

		Funds in ETH	
	Smart Contract	**Received**	**Balance**
1	N/A (Congress Contract) `0x3de0c040705d50d62d1c36bde0ccbad20606515a`	5 028	5 010 ($ 1 918 k)
2	Unicorn Token (Congress Contract) `0xfb6916095ca1df60bb79ce92ce3ea74c37c5d359`	5 891	4 595 ($ 1 760 k)
3	HONG / hongcoin `0x9fa8fa61a10ff892e4ebceb7f4e0fc684c2ce0a9`	3 936	1 003 ($ 384 k)
4	Dogecoin-Ethereum Bounty `0xdbf03b407c01e7cd3cbea99509d93f8dddc8c6fb`	6 592	597 ($ 228 k)

In Figure 2, we compare the complexity of voting methods to the number of method calls. The required gas (on the x axis) is an indicator of the computational complexity. We grouped gas values in buckets of $100 \cdot 10^3$ gas. The consumed gas ranges from 18 120 gas to a maximum of 4 442 268 gas with an average of 82 431 gas. The figure also shows that most voting method calls consume between 100 000 and 200 000 gas (mind the log scale).

3.3 Acquired Funds

Many smart contracts combine one way or another voting with the management of funds. In Table 2, we therefore show the top four deployed voting smart contracts with respect to their funds. We differentiate between the overall received funds and their current balance (as of October 30, 2020). For example, the Unicorn Token uses the Ethereum Foundation DAO Congress contract that allows members to deposit ETH and submit proposals for fundraising; the other members then can vote if the proposal is accepted. After the voting period ends and a pre-defined quorum accepted the proposal, the ETH will be transferred to the proposer automatically.

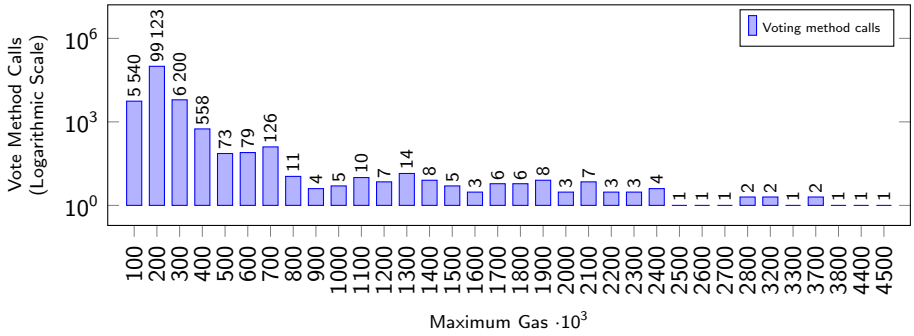

Fig. 2. Complexity of voting methods (measured in gas) in comparison to the number of calls (in total 110 361 calls).

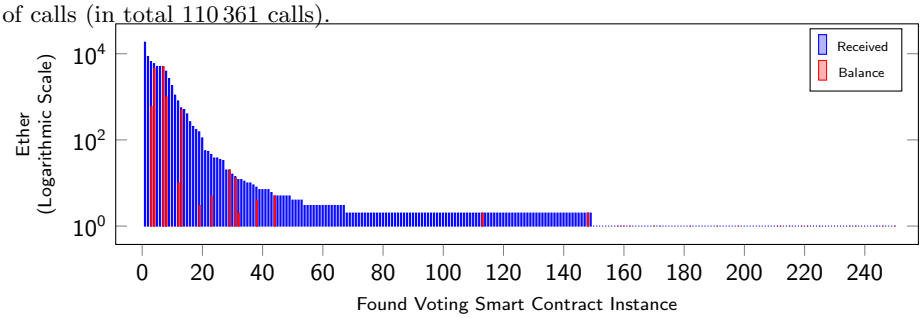

Fig. 3. Received and current balance of ether per voting smart contract, limited to 247 of 3 173 smart contracts in total (as of 2020-10-30).

In Figure 3, we show the distribution of funds (limited to 247 of 3 173 voting smart contracts which have received ETH). We can clearly observe a long tail distribution (log scale). However, many of the originally acquired funds are already withdrawn. From the overall received funds, 0.05 % are still deposited. That is, all analyzed voting smart contracts together have a balance of more than 11 941 ETH, which equals more than 4.8 million USD[8]. The amount of acquired funds can be considered an indicator for the relevance of on-chain voting.

3.4 Trend

In order to get an understanding of the trend, we analyzed the transactions as a time series over the past five years since Ethereum's release in 2015. We particularly focus on the interest and relevance of votings in Ethereum over time.

In Figure 4, we show the number of voting method calls (left y axis) as well as the number of deployed smart contracts related to voting (right y axis). Once deployed, smart contracts remain active and are not counted again in the following years, i.e., the figure shows deployment of new smart contracts. In

[8] Exchange rate at the time of writing was 407 USD per ETH (source: `coinbase.com`)

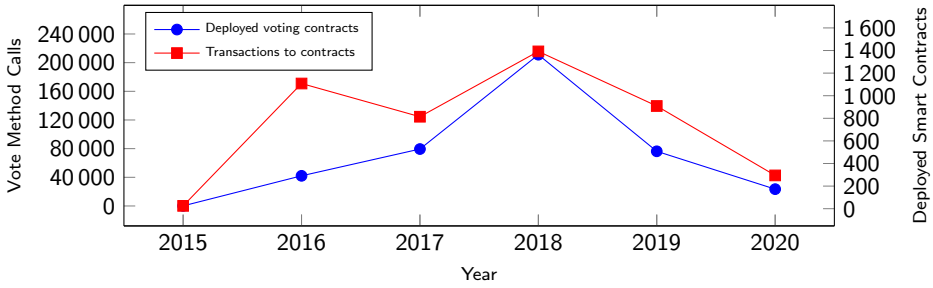

Fig. 4. Number of newly deployed voting smart contracts and transactions to them by year (2015-09-06 – 2020-10-30).

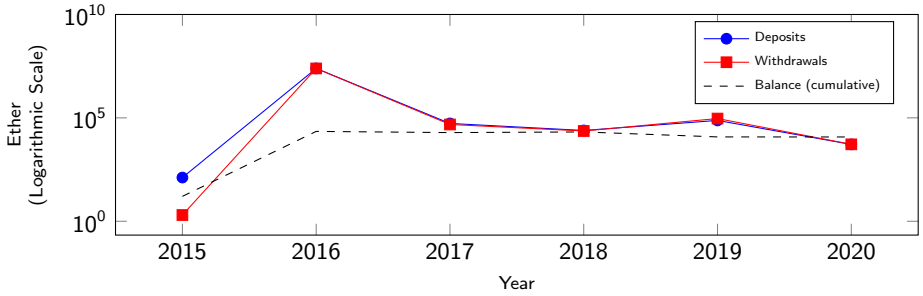

Fig. 5. Ether deposits and withdrawals to voting smart contracts and corresponding balances of voting smart contracts per year (2015-09-06 – 2020-10-30).

addition, we analyzed the deposits, withdrawals, and corresponding balances of each voting smart contract over time, which are shown in Figure 5.

We generally observe that with the debut of the DAO [11] in 2016, the number of smart contracts with voting functionalities as well as the number of transactions that interact with voting contracts increases with a peak in 2018. After 2018, we observe a decline of both metrics. While the trend might suggest a decline in interest, the balances remain stable over time. Upon closer examination, comparing Figure 3 and Figure 5, the total balance in 2020 is almost entirely contributed by the top five voting contracts (with more than 500 ETH). That is, while previously the balances were distributed over many smart contracts, we can infer that funds are more centralized now. We conclude that the dynamics and interactions of voting smart contracts declined over time, but on-chain voting has in terms of funding still a relevance.

4 Feasibility Analysis

In the following, we present a feasibility analysis of on-chain voting. In particular, we analyze scalability limitations using a model-based analysis as well as an empirical analysis based on historical blockchain data.

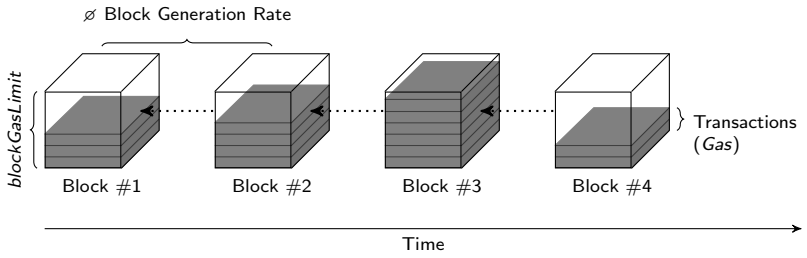

Fig. 6. Blockchain partially filled with transactions, leaving residual gas.

4.1 Block Capacities

One of the central scalability parameters is the maximum number of transactions per block interval, i.e., transaction throughput, which eventually also limits the possible number of votes. Ethereum aims for a block generation rate of 15 seconds and continuously allows miners to agree on a *block gas limit* [27] that limits the size of new blocks. The notion of *gas* was introduced to measure computational complexity of transactions. Ethereum accordingly charges transaction fees based on the transaction's complexity. The sender of a transaction sets a price in ether (ETH), which determines the amount she is willing to pay per computational unit, i.e., the *gas price*.

Depending on the number of transactions per block and their complexity, transactions might not make use of the available block gas limit and leave *residual gas*. In Figure 6, we visualize the concept of the gas consumption and residual gas. The residual gas determines the space for additional transactions on top of the baseline activities of Ethereum. Later, we make use of the notion of residual gas to evaluate feasibility and scale of on-chain voting.

4.2 Model-based Scalability Analysis

Our analysis is based on overly optimistic model-based assumptions to reveal upper limits, which enables us to make fundamental statements on the (in)feasibility of on-chain voting. To this end, we start with a number of votes μ that we would like to cast. We are then interested in the number of blocks n that are necessary to cast μ votes. Given the block generation rate, we can approximate the time it takes to mine n blocks, which we denote with Δ. For a block i with a $blockGasLimit(i)$ and a certain $gasCost$ per vote, we can calculate the maximum number of votes per block by $blockGasLimit(i)/gasCost$.

Based on our blockchain analysis results from Section 3, we evaluate two different scales of voting. Since our measurements show that most voting methods were called between 2 k–7 k times, we consider $\mu = 2\,000$ to be a small-scale voting, and $\mu = 100\,000$ to represent future large-scale votings. Moreover, we introduce three on-chain voting "schemes", which are either overly simple or taken from our previous analysis. Please note that these simple voting schemes are not meant to facilitate general voting principles, e.g., anonymity and secrecy.

Table 3. Required blocks n and duration Δ [HH:MM] for different voting implementations; median and median absolute deviation (MAD) are based on residual block capacities (monthly intervals between 2015-12-28 and 2020-10-30).

	Implementation	Blocks n			Duration Δ		
		Model	Median	MAD	Model	Median	MAD
Small-scale	Ethereum Naïve	4	18	5	00:01	00:04	00:02
	Ethereum Minimal Voting	7	37	14	00:02	00:09	00:05
	Ethereum *The DAO*	25	118	43	00:07	00:30	00:13
	Bitcoin Naïve	1	3	2	00:10	00:29	00:29
	Dash	1	1	0	00:02	00:02	00:00
Large-scale	Ethereum Naïve	175	783	299	00:44	03:24	01:22
	Ethereum Minimal Voting	350	1 634	677	01:28	06:46	02:56
	Ethereum *The DAO*	1 250	7 320	3 232	05:13	33:46	14:19
	Bitcoin Naïve	3	25	20	00:30	05:12	04:38
	Dash	10	10	0	00:25	00:21	00:05

The *naïve voting* provides different addresses, each representing a voting option. Voters can transfer coins to the respective address until the voting ends, where the balances determine the final voting result. This naïve approach can basically be implemented in every cryptocurrency. In Ethereum, the gas costs are 21 000 gas.

The *minimal voting* uses a smart contract for counting votes. To this end, we implemented a synthetic voting smart contract that only consists of a single method for counting votes (available in our Github repository). We are aware, though, that the Solidity compiler does not generate perfectly optimized bytecode. While an optimized voting smart contract with a completely assembly-style built bytecode would need less gas, we consider the Solidity compiler the most prevalent way to compile smart contract code. After deployment, the minimal voting requires at least 41 897 gas per method call.

For the purpose of more realistic statements, we also analyzed the median gas costs of votes to *the DAO*. To this end, we used our analysis pipeline described in the previous section, which yields 150 k gas per DAO voting call. As expected, this is more complex than our minimal voting as it also manages funds and quorum regulations.

In Table 3, we show the minimum duration of small-scale and large-scale votings for the various voting schemes (see "Model" columns). For Ethereum, we assumed a block gas limit of $12 \cdot 10^6$ gas and a block generation rate of 15 seconds. For comparability, we also included the naïve voting for Bitcoin and Dash, which we discuss later in Section 5. Based on this initial evaluation, we can expect that small-scale on-chain voting is generally feasible in reasonable bounds. At the same time, large-scale votings require under idealistic circumstances more than four hours for the naïve voting scheme, or even about 34 h for the DAO voting smart contract.

4.3 Residual Capacities Analysis

In the following, we enrich our model-based evaluation with historic blockchain data to determine the residual gas limits in Ethereum. This approach provides a more realistic assessment of limitations. More specifically, we define $\mathit{residualGas}(i) = \mathit{blockGasLimit}(i) - \mathit{usedGas}(i)$ for a block i. Please note that in Ethereum the block gas limit is block specific and changes over time. The residual gas is therefore determined by the used gas at a certain point in time.

In Table 3, we show the median number of blocks n as well as the duration Δ for historic data in addition to our model-based evaluation. We calculated n and Δ starting with the last mined block of 2020-10-30 and repeated the process for each preceding month until the genesis block of Ethereum (2015-07-30). In general, our measurements yield values under the (unlikely) condition that all voters submit their votes in a perfectly aligned and coordinated order. We use this approach to provide an (optimistic) understanding for the minimum gas needed to deploy and cast a single vote. Since we repeated the evaluation multiple times by shifting starting points in monthly intervals, we present the median absolute deviation (MAD).

The results show that simple small-scale and large-scale voting yield reasonable performance with approx. 30 minutes or less for 2 k votes, and between 30–90 minutes for 100 k votes. The exception is the more complex DAO implementation, which takes more than 5 hours.

4.4 Economic Analysis

Since gas cost can be directly translated to ETH, we can also estimate the economic efficiency of on-chain voting. As a first impression, we consider a median gas price 2.0 $\frac{Gwei}{Gas}$ (SD = 5.92) for the 121 980 voting method calls from our data set. We used an exchange rate of 407 USD per ETH as before. Hence, we can approximate the price of a vote for our minimal voting scheme that approximately yields 0.03 USD per vote. For more realistic gas cost, i.e., the most called voting methods require between 100–200 gas, our price approximation ranges between 0.08 USD and 0.16 USD per vote.

Voting costs are a relevant factor for high reachability and inclusive participation. While fees for casting a vote might serve as Sybil protection, they might also deter voters. In general, fees set a higher participation threshold. In order to maximize participation, transaction costs should be as low as possible for submitting votes—or just not be charged, at all. Unfortunately, smart contracts in Ethereum are not able to pay the transaction fees for the senders, e.g., for calling chosen voting methods. It is possible to implement smart contracts that refund transaction fees within the same transaction, but it still requires voters to own initial ETH for paying the transaction fee in advance. Voters who do not own any ETH hence face a greater hurdle to participate.

Interestingly enough, we want to point out an approach that is able to store and release gas to cover some of the gas costs itself. Projects like the GasToken[9]

[9] `https://github.com/projectchicago/gastoken`

exploit gas reserving opcodes (i.e., `SSTORE` and `CREATE/SELFDESTRUCT`) for saving gas when the gas price is low and releasing it when gas is more expensive. Unfortunately, releasing reserved gas requires gas itself. That is, the transaction costs can be reduced but not covered completely, which leaves us back to the original problem that voters need an initial ETH fund. For enabling future-oriented use cases that require broad involvement, e.g., participatory budgeting or crowd funding, we believe new solutions are required to open on-chain voting.

5 Voting Beyond Ethereum

In the following, we consider other well-established cryptocurrencies, namely Bitcoin [17] and Dash [6], that can also be used for voting one way or another.

5.1 Bitcoin

Several proposals for Bitcoin-based voting exist [3, 25, 28]. Unfortunately, due do the lack of a full-fledged scripting language, Bitcoin heavily relies on external infrastructure to conduct votings, which makes it difficult to inspect the blockchain and reliably extract information with respect to voting. While we have found indications for on-chain voting, infrastructures have been shut down and therefore prevent analysis. Regardless, it is worth mentioning that Bitcoin miners implement voting functionality directly in the blockchain protocol to agree on improvement proposals.[10]

We can however assume that voting would have at least the same transaction requirements (w.r.t. transaction size and cost) as transferring coins. On this basis, we analyze residual transaction capacities of past blocks and derive the maximum of possible votes over that time span. To this end, we need to consider the specifics and changes of the *segregated witness* proposal,[11] which tackles signature malleability issues and therefore separates signature data from the transaction's hashes. As a result, the maximum block size is then limited by the notion of *block weight*, i.e., $\textit{block weight} < 4 \cdot 10^6$, which corresponds approximately to a block size of 4 MB. A standard Bitcoin transaction for transferring coins from one address to another (P2WSH) with segregated witness (inputs and outputs) requires a block weight of approximately 110 (median over all corresponding transactions until Oct 2020 with a standard deviation of 0.069). Other parameters include a target block generation rate of 10 minutes.

Evaluation. Similar to our Ethereum analysis, we analyzed Bitcoin for small-scale and large-scale scenarios with minimal transaction weights, which corresponds to our naïve voting implementation. In addition to a model-based evaluation, we also investigated the residual block capacities.

Table 3 shows the minimum amount of blocks as well as the time span it would take to cast μ votes. We assumed a transaction weight of 110 per vote. While

[10] `https://github.com/bitcoin/bips/blob/master/bip-0009.mediawiki`
[11] `https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki`

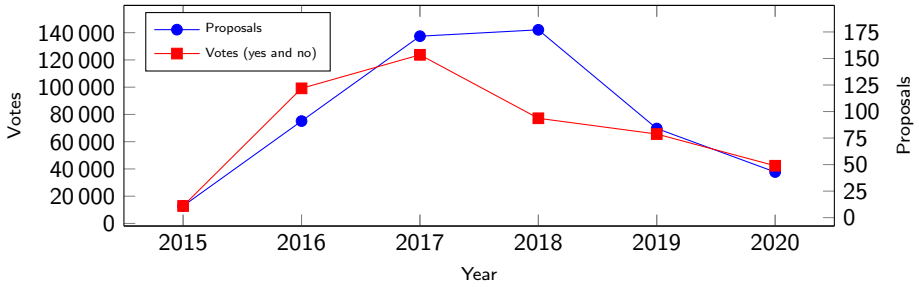

Fig. 7. Number of Dash governance proposals and the number of votes, grouped by year (2015-08-27– 2020-10-26).

Ethereum requires at least 4 minutes, Bitcoin requires 29 minutes for small-scale votings. Please note that Bitcoin indicates very high MAD for n and Δ. Hence, Bitcoin's residual capacities fluctuate significantly compared to Ethereum, which makes it more difficult to make predictions. For large-scale voting, Bitcoin requires significantly less blocks (due to the larger block size) and despite its slower block generation rate is faster than Ethereum.

5.2 Dash Governance Platform Analysis

Dash [6] was released in 2014, initially named *Xcoin* and later *Darkcoin*. Dash does not support smart contracts in the same way as Ethereum, but implements dedicated governance mechanisms directly in its protocols. During the mining process new coins will be split and distributed over three stakeholders: master nodes and miners receive each 40 %, and the remaining 20 % go to Dash's Decentralized Governance by Blockchain (DGBB) funding platform. Master nodes then can vote on public proposals for distributing the collected funds.

The Dash Governance Platform (DGP) is natively implemented in Dash's application protocols and therefore can be monitored by all nodes that have joined the network. After a pre-defined voting phase, the number of yes-votes minus the no-votes must exceed 10 % of the total number of master nodes for a proposal to pass. Otherwise, the proposal will be rejected. For better accessibility, DashCentral[12] provides an overview of all proposals and a public API.

Evaluation. In order to get a first impression, we analyzed 577 proposals between 2015-08-27 and 2020-10-26. During that time, 379 proposals were funded. In Figure 7, we show the total number of votes and corresponding number of proposals per year. Dash's governance proposals started at the same year as the first voting smart contracts on the Ethereum blockchain in 2015. Dash shows an increase and peak of newly created proposals and votings between 2016 and 2018, and similar to Ethereum, a steady decrease of interest afterwards. Dash's

[12] `https://www.dashcentral.org`

number of proposals at the peak is approximately 8 times is lower compared to Ethereum (c.f. Figure 4). Note that the analysis of Dash is more precise and does not suppress any false-positives, which means that the difference to Ethereum is probably even higher. The number of votes at peak times is approximately 1–3 times smaller, when compared to Ethereum. All successful proposals collected 131 453 DASH, which equals approximately 14.7 million USD according to the corresponding exchange rates at the time of funding.[13] Even though the presented votings were not conducted on-chain, the blockchain's protocol automatically pays out fundings with Dash's cryptocurrency and therefore supports the role of on-chain voting.

Additionally, we evaluated the residual capacities of the Dash blockchain. While Dash is based on Bitcoin, it does not support segregated witness and aims for a block generation rate of 2.5 minutes with a maximum block size of 2 MB. As shown in Table 3, Dash does not have such a high transaction load as Bitcoin or Ethereum, which directly leads to high residual capacities and therefore better performance for small-scale and large-scale voting. Our measurements show even better results than our model approximation, because the proof-of-work consensus generated new blocks faster than expected. We nevertheless would expect higher durations with the same general load, i.e., residual capacity, as in Bitcoin.

6 Conclusion

In this paper, we have shown that on-chain voting has become a relevant use case in Ethereum, most often, to collectively manage funds. To this end, we presented our blockchain analysis toolchain, that we used to identify and analyze voting smart contracts with respect to their popularity, complexity, and funds. On the one hand, our benchmark of transactions to voting smart contracts and their respective fundings confirm a high relevance. On the other hand, we observed a trend of centralization due to the popularity of DAO contracts.

We further used these insights to assess the feasibility of future large-scale voting on blockchains. Therefore, we also evaluated other well-established blockchains, i.e., Bitcoin and Dash. While small-scale voting scenarios seem feasible on all analyzed blockchains, large-scale voting suffers from severe scalability issues. Although our model-based calculations indicate that large-scale votings can theoretically be conducted in reasonable times under perfect conditions, our measurements on well-established public blockchains show that minimum durations increase significantly due to the limited transaction throughput.

Despite all the flaws of blockchain-based voting, We have shown that on-chain voting has a relevance, e.g., for governance aspects of blockchains. We therefore believe that improving on-chain voting schemes with respect to security, privacy, inclusiveness, and fairness is still necessary and relevant at the same time.

[13] `https://coinmarketcap.com/en/currencies/dash/historical-data/` (Accessed: 2020-11-16)

Bibliography

[1] Atzei, N., Bartoletti, M., Cimoli, T.: A Survey of Attacks on Ethereum Smart Contracts (SoK). International Conference on Principles of Security and Trust, Springer (2017)

[2] Béres, F., Seres, I.A., Benczúr, A.A., Quintyne-Collins, M.: Blockchain is Watching You: Profiling and Deanonymizing Ethereum Users. CoRR (2020)

[3] Bistarelli, S., Mantilacci, M., Santancini, P., Santini, F.: An End-to-end Voting-system Based on Bitcoin. In: SAC, ACM (2017)

[4] Croman, K., Decker, C., Eyal, I., Gencer, A.E., Juels, A., Kosba, A.E., Miller, A., Saxena, P., Shi, E., Sirer, E.G., Song, D., Wattenhofer, R.: On Scaling Decentralized Blockchains - (A Position Paper). In: Financial Cryptography Workshops, Lecture Notes in Computer Science, Springer (2016)

[5] Dimitriou, T.: Efficient, Coercion-free and Universally Verifiable Blockchain-based Voting. Computer Networks (2020)

[6] Duffield, E., Diaz, D.: Dash: A Payments-Focused Cryptocurrency (2018), URL `https://github.com/dashpay/dash/wiki/Whitepaper`, accessed: 2020-10-26

[7] Fröwis, M., Fuchs, A., Böhme, R.: Detecting Token Systems on Ethereum. In: International Conference on Financial Cryptography and Data Security, Springer (2019)

[8] Heiberg, S., Kubjas, I., Siim, J., Willemson, J.: On Trade-offs of Applying Block Chains for Electronic Voting Bulletin Boards. E-Vote-ID (2018)

[9] Henry, R., Herzberg, A., Kate, A.: Blockchain Access Privacy: Challenges and Directions. IEEE Security & Privacy (2018)

[10] Hjalmarsson, F.P., Hreioarsson, G.K., Hamdaqa, M., Hjálmtýsson, G.: Blockchain-based E-Voting System. In: IEEE CLOUD (2018)

[11] Jentzsch, C.: Decentralized Autonomous Organization to Automate Governance. White paper (2016)

[12] Killer, C., Rodrigues, B., Matile, R., Scheid, E.J., Stiller, B.: Design and Implementation of Cast-as-Intended Verifiability for a Blockchain-based Voting System. In: SAC, ACM (2020)

[13] Kroll, J.A., Davey, I.C., Felten, E.W.: The Economics of Bitcoin Mining, or Bitcoin in the Presence of Adversaries. In: Proceedings of WEIS (2013)

[14] Kshetri, N., Voas, J.M.: Blockchain-Enabled E-Voting. IEEE Software (2018)

[15] Matzutt, R., Hiller, J., Henze, M., Ziegeldorf, J.H., Müllmann, D., Hohlfeld, O., Wehrle, K.: A Quantitative Analysis of the Impact of Arbitrary Blockchain Content on Bitcoin. In: Financial Cryptography, Lecture Notes in Computer Science, Springer (2018)

[16] McCorry, P., Shahandashti, S.F., Hao, F.: A Smart Contract for Boardroom Voting with Maximum Voter Privacy. In: International Conference on Financial Cryptography and Data Security, Springer (2017)

[17] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)

[18] National Academies of Sciences, Engineering, and Medicine and others: Securing the Vote: Protecting American Democracy pp. 103–105 (2018)

[19] Park, S., Specter, M., Narula, N., Rivest, R.L.: Going from Bad to Worse: From Internet Voting to Blockchain Voting (2020), URL `https://people.csail.mit.edu/rivest/pubs/PSNR20.pdf`, accessed: 2020-11-24
[20] Pinna, A., Ibba, S., Baralla, G., Tonelli, R., Marchesi, M.: A Massive Analysis of Ethereum Smart Contracts Empirical Study and Code Metrics. IEEE Access (2019)
[21] Reibel, P., Yousaf, H., Meiklejohn, S.: Short Paper: An Exploration of Code Diversity in the Cryptocurrency Landscape. In: Financial Cryptography, Lecture Notes in Computer Science, Springer (2019)
[22] Reid, F., Harrigan, M.: An Analysis of Anonymity in the Bitcoin System. In: SocialCom/PASSAT, IEEE Computer Society (2011)
[23] Specter, M.A., Koppel, J., Weitzner, D.: The Ballot is Busted Before the Blockchain: A Security Analysis of Voatz, the First Internet Voting Application Used in U.S. Federal Elections. In: 29th USENIX Security Symposium (2020)
[24] Szabo, N.: Formalizing and Securing Relationships on Public Networks. First Monday, Vol. 2, Nr. 9 (1997)
[25] Tian, H., Fu, L., He, J.: A Simpler Bitcoin Voting Protocol. In: Int. Conference on Information Security and Cryptology, Springer (2017)
[26] Victor, F., Lüders, B.K.: Measuring Ethereum-based ERC20 Token Networks. In: Financial Cryptography, Lecture Notes in Computer Science, Springer (2019)
[27] Wood, G.: Ethereum: A Secure Decentralised Generalised Transaction Ledger, Byzantium Revision 7E819EC (2019-10-20)
[28] Zhao, Z., Chan, T.H.H.: How to Vote Privately using Bitcoin. In: Int. Conference on Information and Communications Security, Springer (2015)