The Referendum Problem in Anonymous Voting for Decentralized Autonomous Organizations

Artem Grigor[1,†], Vincenzo Iovino[1,*,†] and Giuseppe Visconti[2,†]

[1]Aragon ZK Research
[2]University of Salerno

Abstract
A natural approach to anonymous voting over Ethereum assumes that there is an off-chain *aggregator* that performs the following task. The aggregator receives valid signatures of YES/NO preferences from eligible voters and uses them to compute a zk-SNARK proof of the fact that the majority of voters have cast a preference for YES or NO. Then, the aggregator sends to the smart contract the zk-SNARK proof, the smart contract verifies the proof and can trigger an action (e.g., a transfer of funds). It is believed that as the zk-SNARK proof guarantees anonymity, the privacy of the voters is preserved by attackers not colluding with the aggregator. Moreover, if the SNARK proof verification is efficient the GAS cost will be independent on the number of participating voters and signatures submitted by voters to the aggregator.

In this paper we show that this naive approach to run referenda over Ethereum can incur severe security problems. We propose both mitigations and hardness results for achieving voting procedures in which the proofs submitted on-chain are either ZK or succinct.

Keywords
blockchain, e-voting, zero-knowledge, consensus

1. Introduction

The main use of Voting on Ethereum network are to facilitate functioning of Decentralized Autonomous Organizations (DAOs). DAOs are members-owned communities without centralized leadership. Most of the DAOs that operate over the Ethereum network and are governed by simple smart contracts. One of the main DAOs' functionalities is to peform on-chain actions such as transfers of funds to an account if a sufficient number of DAO's members vote for that.

One of the main types of voting procedures for DAOs is the *referendum*. In a referendum, a transfer (or any other on-chain action) is accepted and executed if there is support from the majority of the voters, where voters can choose between supporting (casting YES options) and opposing (casting NO option). Moreover, it is common for each voter to have a specific weight based on the amount of their funds (token). For simplicity, hereafter we will suppose that each voter has the same weight.

Recently, it is becoming more and more important for DAOs to be able to hold anonymous referendum.[1] However, to tally such referendum a large amount of GAS needs to be consumed

5th Distributed Ledger Technology Workshop (DLT 2023), May 25–26, 2023, Bologna, Italy
*Corresponding author.
†These authors contributed equally.
artem@aragon.org (A. Grigor); vincenzo@aragon.org (V. Iovino); peppevisconti93@gmail.com (G. Visconti)

on the Ethereum network, making voting financially impractical.[1] To address this problem, some researchers and engineers (e.g., [2, 1]) are proposing solutions that minimise the GAS consumption on-chain as follows. The voting procedure is done off Ethereum chain (e.g., on another Blockchain) where voters sign and cast YES/NO options without incurring any GAS costs, and then an *aggregator* computes a non-interactive zero-knowledge (NIZK, in short) proof π of the correct result of the election. The NIZK security guarantees that if the proof is accepted by the corresponding NIZK Verifier, the result is correct. This allows to only send result and its NIZK proof to the Verifier smart contract on the Ethereum network, thus reducing and transferring the GAS costs from voters to the aggregator.

More precisely, the system works as follows. The public-keys (PKs) of the D eligible voters are arranged in a Merkle Tree [3] whose root C is called the *census*. Each voter signs his own preference using their own PK and sends it to an aggregator. Suppose that k_0, k_1 voters submit a NO and YES preference respectively. The aggregator creates a NIZK proof to prove knowledge of k_0 signatures for NO and k_1 signatures for YES, where each signature is verified to correspond to one of the PKs in the Merkle tree with census C. Moreover, the proof guarantees that the signatures come from distinct eligible voters.

Using general-purpose succinct NIZK proofs (called *SNARKs*) [4, 5], the scheme can be further improved to minimize GAS consumption on Ethereum. For instance, in the Groth's SNARK [4] the proof can be verified using 3 pairings and t exponentiations where t is the number of elements of the public statement. Which would allow to have GAS cost independent of the referendum size.

Recursive proofs. We motivated the problem in the context of recorded voting in which the aggregators can completely break the privacy of the voters. This is without loss of generality, in fact our analysis and results also applies to solutions in which the aggregators receive instead anonymous proofs of membership (rather than plain signatures). Restricting the attention to non-anonymous solutions makes our results stronger. Currently many engineers and researchers are actively working on recursive proofs [6, 7] and one of the natural applications of them is to minimize the GAS costs of anonymous voting solutions [1]. Our results show that while recursive proofs can still be useful to minimize GAS costs of anonymous voting solutions for web3, it is not clear how to use them to achieve constant GAS time limit.

The problem. At first sight it may seem that zk-SNARKs allow to minimize GAS cost while preserving verifiability and privacy. However, this is because we overlook the following problem:

> What if there are several proofs with conflicting result received by the smart contract? In this case, which result should be accepted by the smart contract?

First of all, we clarify why there can be no single aggregator authorized by the contract to submit proofs, which allows us to reason that we could have multiple conflicting proofs submitted to the contract. In the rest of this paper we will investigate how to manage these conflicting proofs.

[1]GAS is a measure of computation cost on Ethereum in abstract units. If multiplied by GAS cost it provides the real cost of the execution of a transaction in Eth units.

Observe that we have two mutually exclusive cases.

- There is a single aggregator authorized by the contract to send data to the smart contract. We show that such case is insecure as the authorized aggregator has the capacity to cheat and thus for the security of our system we need to consider an alternative case.
  In order to see this, let us consider a case when there is at least one valid signature for either YES or NO. Let P be the authorized aggregator. Then P can simply submit an arbitrary result. For instance, despite receiving (k_1, k_2) signatures for YES and NO, the P can send to the contract proof that there are (k_1', k_2') signatures for YES and NO, where $k_1' \leq k_1, k_2' \leq k_2$. Which gives P the ability to sway results in arbitrary directions in S_2. So, using SNARK proofs does not guarantee the correctness of the submitted election results by P.
  This implies that we still need to trust the authorised aggregator P.
- There is no single aggregator authorized by the contract to submit proofs. In this case, it can be that two aggregators, P_1 and P_2, submit different proofs. This can be achieved by both P_1 and P_2 omitting different voters' ballots from their proofs. This implies that we might end up in a situation where there are multiple conflicting proofs submitted to the contract, which we will investigate further in the paper.

2. The referendum problem in web3 voting for DAOs

Suppose the eligible voters are A, V, R, C, M, B, J, S.

In the following we suppose that there are different aggregators running on nodes $N_1, N_2, \ldots$.

Suppose that the smart contract receives from node N_1 a SNARK proof π_1 of the claim that the result of the referendum is $(3, 2)$, that is 3 votes for YES and 2 for NO signed by different eligible voters in the census.[2]

The knowledge soundness of the SNARK guarantees that there actually exist 3 signatures for YES and 2 for NO and that those signatures are from different eligible voters. Suppose that these signatures for YES correspond to (A, V, R) and the ones for NO to (C, M), precisely let us suppose that the aggregator running on node N_1 produced the proof π_1 after having received 3 YES signatures from (A, V, R) and 2 NO signatures from (C, M).

The smart contract cannot accept the first incoming proof: indeed, it might be the case that node N_1 received instead 3 signatures for YES from (A, V, R) and 5 signatures for NO from (C, M, B, J, S) and N_1 did not take in account the signatures of (B, J, S) for the computation of the proof π_1. So the smart contract actually needs to wait for more incoming proofs[3] $\pi_2, ..., \pi_n$ for some $n \geq 1$ in a given window of time. Then, from the n received proofs the smart contract must decide the result of the election (YES, NO or TIE).

Our setting and threat model. In our setting nodes are computers over the world that can submit information to the smart contract. Since the smart contract has no authentication, the

[2]Hereafter, we implicitly suppose that the claim is with respect to a known census but we omit this detail for simplicity.

[3]Hereafter, we will sometimes assume that a proof is associated with a claim of the form (k_0, k_1), for two integers $k_1, k_2 \geq 0$.

smart contract may not know which information comes from which node. In our setting, we assume that an attacker has the following capabilities:

1. The attacker can control multiple nodes. For each controlled node, the attacker waits for signatures from voters (both honest and dishonest voters) and can use them to generate proofs to send to the smart contract.
2. The attacker can observe signatures sent by honest voters to other nodes, not under the attacker's control, and replay them to the attackers' controlled nodes. That is the attacker can do copy-and-paste attacks.
3. The attacker is capable of corrupting a subset of voters.

Moreover, we clarify that means for a node to be honest. We provide this notion for respectively two different protocol classes. One class of protocols prescribes that honest voters must send their votes to a fixed node (or set of nodes). In this case, with a slight abuse of notation, we can say that this node specified by the protocol is the honest node, meaning that it is the node to which honest voters are required to send their votes (if a voter does not send his own vote to that node, such voter acts maliciously). The other class of protocols instead just gives freedom to the voters to either send their own vote to the smart contract directly (in this case the voter acts as a node) or use any other node over the network as an aggregator. In this case, a node is honest if it acts as an aggregator prescribed by the protocol. In this context, a dishonest node could for example compute proofs with respect to a smaller subset of signatures, possibly removing signatures to its like.

Policy and properties. We term *policy* the procedure that the smart contract implements to decide the result of the election (YES or NO or TIE) based on the n received proofs.

A good policy could be one that satisfies the following property P_1. Before defining P_1, we need some definition and notational conventions. Consider the set S_Y (S_N) of voters who cast a vote for YES (NO) to some node (any computer in the world). Then S_Y (resp. S_N) includes v iff a voter v sent a YES (resp. NO) to some (possibly dishonest) node. Let $\mathsf{Bad} = S_Y \cap S_N$, that is the set of *bad* voters who cast a vote both for YES and NO, and let us denote by $S'_Y = S_Y - \mathsf{Bad}$ and $S'_N = S_N - \mathsf{Bad}$ the new sets in which we remove the bad voters. So for example, if voter v is the only one who sent both YES and NO votes to different nodes, then S'_Y and S'_N are identical to S_Y and S_N except that they do not include v. Then, the property P_1 is formalized as follows.

- Property P_1: a policy is good if the decision output by the policy is equal to YES if $|S'_Y| > |S'_N|$, is equal to NO if $|S'_N| > |S'_Y|$ and is equal to TIE if $|S'_Y| = |S'_N|$.

What property P_1 entails in the real world. As stated, property P_1 may seem nonsensical or unachievable: indeed, requiring that the decision output by the policy should be related to all votes recorded on any node over the world seems too strong. The justification of the property is that we can instead assume that if a vote was recorded over a node N, the node N will *commit* to this vote on-chain by sending a proof computed with a set of signatures that includes the one corresponding to this vote (and possibly others). Observe that, in order to make this reasoning sound, we must assume that our proofs are binding in the following sense. Proof for a given claim is binding if it can be opened (by revealing the random coins and the witness used to produce it) only with a unique set of signatures compatible with that claim.[4] Henceforth, we will assume that the proofs computed by honest nodes have such property.

Therefore, we can assume that the sets S_Y and S_N have been committed to the smart contract and so it is meaningful to require that the smart contract should accept the decision given by property P_1 that in this case would be relative to votes that are recorded on the smart contract itself (and not on arbitrary nodes spread over the world).

Finally, a natural question is the following: why in protocols in which honest voters have a fixed node (or set of nodes) to which to send their votes, property P_1 should be regarded as good? More precisely, if there is an honest node, why should be the decision based on votes recorded outside such an honest node? The reason is that the smart contract has no authentication and thus has no possibility of establishing who is the honest node. For this reason, property P_1 seems reasonable even for this class of protocols. Later on, we will discuss an alternative property that is restricted to only this class of protocols.

Unsatisfactory solution 1. One can think that the following policy is somehow "good": from the many accepted proofs consider the one with the highest number of total votes and outputs the result given in that proof.

Consider as before a node N_1 that receives 3 YES votes from (A, V, R) and 5 NO votes from (C, M, B, J, S) and computes a proof π_1 for the claim $(3, 2)$ from the YES votes of (A, V, R) and the NO votes of (C, M) without taking in account the NO signatures of (B, J, S) for the computation of π_1. Consider now a node N_2 that received YES votes from (A, V, R, C) and NO votes from (M, B, J) and sends the corresponding proof π_2 of the result $(4, 3)$ to the smart contract. Observe that C has been malicious because he sent to N_1 a NO vote and to N_2 a YES vote. Suppose that a third node N_3 received YES votes from (A) and NO from (B, J, S).

Node N_3 based on these signatures sends to the smart contract a proof π_3 of the result $(1, 3)$. Suppose that the time window to submit proofs ended and no more proofs can be sent to the smart contract that must compute the final decision based on all proofs received until now. The smart contract received the following pairs of proofs and claims:

- Proof π_1 for $(3, 2)$ from N_1 computed with YES votes of (A, V, R) and NO votes of (C, M).
- Proof π_2 for $(4, 3)$ from N_2 computed with YES votes of (A, V, R, C) and NO votes of (M, B, J).
- Proof π_3 for $(1, 3)$ from N_3 computed with YES vote of (A) and NO votes of (B, J, S).

So the proof with the highest number of votes is the second and this corresponds to a YES result that will be the result announced by the smart contract. Observe that only N_1 has been malicious in removing the NO signatures of (B, J, S). The other nodes acted honestly based on the signatures they received. Consider what each voter did:

- V, R: sent a YES vote to nodes N_1, N_2 and nothing else.
- M: sent a NO vote to nodes N_1, N_2 and nothing else.

[4]SNARK proofs can be combined with commitments to satisfy this property.

- A: sent a YES vote to all nodes N_1, N_2, N_3 and nothing else.
- C: sent a NO vote to node N_1 and a YES vote to node N_2 and nothing else.
- S: sent a NO vote to both N_1 and N_3 and nothing else.
- B, J: sent a NO vote to all nodes N_1, N_2, N_3 and nothing else.

Thus, removing the bad voter C that voted both for YES and NO, we have that (A, V, R) voted only for YES (possibly replicating the votes to different nodes) and the voters (M, B, J, S) voted only for NO (possibly with replication). Therefore, the actual result should be NO according to property P_1 but the policy outputs YES.

In the previous counter-example, we were implicitly assuming that honest voters have no fixed set of nodes to which to send their votes. However, the counter-example can be also requested for protocols in which the honest voters have a fixed node (or set of nodes) to which to send their votes. In fact, suppose that honest voters are prescribed to send their votes to node N_3. In that case, whenever before we said that, for instance, A sent a YES vote to all nodes, we actually mean that A honestly sent the YES vote to just node N_3. Then attacker copied this vote and replicated it on nodes N_1 and N_2. Whenever we said that, for instance, M sent a NO vote to nodes N_1, N_2 we actually mean that the attacker corrupts M so, having access to M's secret key, the attacker can generate any signatures on behalf of M and send them to N_1, N_2. So, in the previous counter-example, we would have that (M, C, V, R) were corrupted and (A, S, B, J) were honest.

Unsatisfactory solution 2. One can think of the following alternative solution to the referendum problem. We restrict attention to protocols in which honest voters send their votes to a single honest node. We provide such an assumption as a set of honest nodes can be for simplicity reduced to a single honest node using consensus algorithms.

Then, we can propose the following alternative property P_2.

- Property P_2: a policy is good if the decision output by the policy corresponds to the result consistent with the signatures received by the trusted node removing bad voters who sent to the trusted node votes for both YES and NO. That is if the trusted node (recall that we are assuming for simplicity that the set of trusted nodes reduces to a single trusted node) received m signatures for YES and n for NO and $m > n$ the result is YES, etc.

Then, one can think that the following policy is good for property P_2: from the many accepted SNARK proofs consider the one with the highest number of total votes and outputs the result given in that proof. Notice that the policy is the same as before but we now request that it satisfy a different property.

We show that this policy is still not good with respect to property P_2. In fact, in the previous counter-example, we can think of node N_3 as being the honest one.

Then, the voters (A, S, B, J) are honest because they never send contradictory results to different nodes. However, we assumed that some of them sent votes not just to node N_3 but also to others (e.g., A sends votes not just to N_3 but also to N_1, N_2).

This can be seen as contradicting the honesty of such voters. This is instead not correct: it could be that such voters have been honest in sending their own signatures of just the node N_3 but the attacker(s) did copy-and-paste attacks to copy such signatures also on other nodes.

So, according to our policy, the smart contract will select the second proof for the result $(4, 3)$ and so will output YES. But, by definition of property P_2 the result would have been the one contained in node N_3 and N_3 received a single signature for YES from (A) and 3 signatures for NO from (S, B, J), so the result should have actually been NO.

Alternative formulation of the previous attack. The previous attack can be reformulated as follows. There is an honest party representing a node (or set of nodes) that implements a public blockchain; in the previous formulation, this is the node N_3.

The voters (A, S, B, J) are honest and in particular, (A) sent a YES signature to the blockchain and the others send a NO signature. The node running this blockchain then sends a proof π for the result $(1, 3)$ to the smart contract (this proof corresponds to proof π_3 in our previous formulation of the attack).

Then, there is an attacker that can corrupt the other eligible voters (V, R, M, C) and read the public blockchain. The attacker can generate proof for the result $(4, 3)$ using YES signatures of (V, R, A, C) and NO signatures of (M, B, J). Note that A, B, J are not corrupted but the attacker can generate signatures for them by just copying their signatures from the public blockchain.

The attacker sends such proof π' to the smart contract (this corresponds to the proof π_2 in the previous formulation of the attack). Now according to the policy (the one that selects the proof with the highest number of total votes), the result YES corresponding to the tally $(4, 3)$ is selected rather than NO corresponding to the tally $(1, 3)$ appearing on the public blockchain.

Is property P_2 preferable to property P_1? One can think that intuitively property P_2 is somehow preferable because it selects a result consistent with a public blockchain whose "truth" is agreed upon by honest parties. Unfortunately, we show that if property P_2 is what we really want, we end up restricting the class of solutions we can adopt to solve the referendum problem.

Imagine that our SNARKs have the following magic aggregation property better explained by example. Suppose that a proof π_1 of the result $(1, 2)$ was computed by YES signature of (A) and NO signatures of (V, M) and that a proof π_2 of the result $(2, 1)$ was computed by YES signatures of (A, M) and NO signature of (S).

Then, from these two proofs, anyone can compute a new proof π of the result $(1, 2)$ corresponding to single YES signature of (A) and NO signatures of (V, S). Note that we removed any signature of M from the counting because M can be seen as a bad voter who voted both YES and NO and we did not count twice A (A might be honest but subject to a copy-and-paste attack). Suppose also that the new proof somehow preserves privacy and is still succinct.

This ideal cryptographic primitive can be seen as a solution to the referendum problem while preserving privacy and succinctness. Indeed, consider the example in the paragraph of the ”Unsatisfactory Solution 1” and suppose that N_2 is the honest node to which honest voters send their votes.

From the 3 proofs π_1, π_2, π_3, anyone can generate a single proof π of the result $(3, 3)$ corresponding to YES signatures of (A, V, R) and NO signatures of (B, J, S) where the NO signatures of M has been removed because M voted both YES and NO.

Then, notice that a policy that uses these aggregatable SNARKs to select the result would output as result TIE. Unfortunately, according to property P_2 the result should instead be YES because the honest node N_2 received 1 signature for YES and 3 signatures for NO.

Therefore we have that either P_2 is not a wished property or this ideal aggregatable SNARK primitive would not help. Observe that this problem would also occur if we considered an ideal aggregatable variant of SNARKs that does not remove bad voters.

Summarizing the problem. The issues analyzed before arise from the fact that the smart contract has no possibility of checking which node is honest and moreover, there is no blockchain that acts as a single source of truth.

It seems perfectly legitimate to assume that a voter can replicate his own vote to different nodes. But even if we considered this as a malicious action to be prohibited, we cannot check that. Indeed, a SNARK proof hides who voted for which preference and from which entity the signatures were collected. So, enforcing that replication is not carried out by any eligible voter seems to clash with privacy and succinctness. Observe that our problem is seemingly a variant of the Byzantine agreement problem.

Recasting. If recasting is allowed, for example for coercion-resistance, we cannot even deem a voter to be bad if he sent contradictory results to different nodes. In that case, it seems even harder to handle the result of the referendum. Since in this note, we attempted to show negative counter-examples on the feasibility of handling referenda with privacy and succinctness, we can for simplicity assume that recasting is not allowed.

3. Impossibility results

3.1. Impossibility for aggregatable NIZK

Hereafter, we assume the reader familiar with the notion of NIZK [8, 4]. Consider the following notion of *aggregatable non-interactive proof.*

Definition 1. *A non-interactive proof system (K, P, V), where K is the algorithm to generate the CRS, P is the prover and V the verifier, is aggregatable if it is endowed with an efficient aggregator algorithm* Agg *that satisfies the following property.*

Let $i \in \{1, 2\}$. In the following let σ be the CRS output of K on the security parameter.

Let π_1 be a proof computed by P with CRS σ and claim that there is a single YES signature and 0 *NO signatures computed with witness consisting of a YES signature of voter i computed with its secret-key* Sk_i.

Let π_2 be a proof computed by P with CRS σ and claim that there is a single YES signature and 0 *NO signatures computed with witness consisting of a YES signature of voter* 1 *computed with its secret-key* Sk_1.

Let π' be the output of $\mathsf{Agg}(\sigma, \pi_1, \pi_2)$.

Let c_1 be the claim that there is a single YES signature and 0 *NO signatures and let c_2 be the claim that there are two YES signatures and* 0 *NO signatures.*

Then the following properties hold.

- *Property F_1. If $i = 1$ then: $V(\sigma, c_1, \pi') = 1$ and $V(\sigma, c_2, \pi') = 0$.*
- *Property F_2. If $i = 2$ then: $V(\sigma, c_2, \pi') = 1$.*

Observe now that if SNARK satisfied the above notion of aggregation then it could be used in the obvious way to guarantee Property 1 discussed in Section 2. In the rest of this section, we prove the following theorem.

Theorem 1. *There exists no aggregatable NIZK.*

Proof 1. *To show that an aggregatable non-interactive proof system cannot be zero-knowledge we now construct an adversary* Adv *against the ZK property. Recall that* Adv *is given access to an oracle O that can be either the honest prover oracle, that we will denote P', or the simulator oracle, that we will denote S'.*

Adv *takes as input a CRS σ and does the following.*

Adv *picks i at random from the set $\{1, 2\}$.* Adv *computes the pairs of public- and secret- keys for voters $1, 2$ and YES signatures σ_1, σ_2 for them.* Adv *computes the census based on the public-keys of the two voters and witness w consisting of σ_i.*

Let c_1 be the claim that there is a single YES signature and 0 NO signatures and let c_2 be the claim that there are two YES signatures and 0 NO signatures.

Adv *asks to the oracle a proof for the claim c_1 and witness w and gets π_1 from the oracle.*

Adv *computes π_2 invoking the prover on the CRS σ, the claim c_1 and the witness σ_1.* Adv *also computes $\pi' \leftarrow \mathsf{Agg}(\sigma, \pi_1, \pi_2)$.*

If $V(\sigma, c_1, \pi') = 1$ and $V(\sigma, c_2, \pi') = 0$, Adv *outputs 1 to denote that he guesses that the oracle was P', outputs 0 otherwise to denote that he guesses that the oracle was S'.*

Analysis of Adv**.**

Let H be the event that Adv *outputs 1. Let E be the event that the oracle is equal to S' and the CRS σ was simulated. Let F be the event that the oracle is P' and CRS σ is honestly computed.*

Then, it holds the following. By property F_1 we have that (A) $\mathrm{Prob}[H|F \wedge i = 1] = 1$, and (B) $\mathrm{Prob}[H|F \wedge i = 2] = 0$.

Since π_1 is independent on i when $O = S'$ then (C) $\mathrm{Prob}[H|E \wedge i = 1] = \mathrm{Prob}[H|E \wedge i = 2]$.

By ZK, $|\mathrm{Prob}[H|E \wedge i = 1] - \mathrm{Prob}[H|F \wedge i = 1]| = n_2$, for some negligible function n_2 of the security parameter and then, by (A), it follows that (A') $\mathrm{Prob}[H|E \wedge i = 1] = 1 + n_1$.

By ZK, $|\mathrm{Prob}[H|E \wedge i = 2] - \mathrm{Prob}[H|F \wedge i = 2]| = n_3$, for some negligible function n_3 of the security parameter and then, by (B), it follows that (B') $\mathrm{Prob}[H|E \wedge i = 2] = n_3$.

Therefore, $(A'), (B'), (C)$ imply $1 + n_1 = n_3$, a contradiction.

On Fully Homomorphic NIZKs of [9]. It may seem that our theorem contradicts the existence of fully homomorphic NIZKs of [9]. However, the latter notion only applies to recursive circuit composition in which the output of two or more circuits is given as input to an upper level circuit; this is weaker than needed by our notion of aggregatable NIZKs.

3.2. A mitigation to the referendum problem: trading privacy for succinctness

The above impossibility result can be somehow bypassed as follows.

Each eligible voter v, beyond a signature, also sends to the node a value of the form $\mathsf{Hash}(\mathsf{Sk}_v, id)$ where id is the identifier of the election and Sk_v is his own secret-key. Each proof is accompanied by two lists, resp. the YES list and the NO list, of values of the form $\mathsf{Hash}(\mathsf{Sk}_v, \mathsf{id})$, wherein the first list includes the values corresponding to voters who cast a preference for YES and in the second list one includes the values corresponding to voters who cast a preference for NO.

So a proof for 2 YES from (A, M) and 2 NO from (C, J) would come with the two lists $(\mathsf{Hash}(\mathsf{Sk}_A, \mathsf{id}), \mathsf{Hash}(\mathsf{Sk}_M, \mathsf{id}))$ and $(Hash(\mathsf{Sk}_C, \mathsf{id}), Hash(\mathsf{Sk}_J, \mathsf{id}))$. Moreover, the two lists are hashed into a digest H and the SNARK circuit needs to check if H corresponds to the two lists and that for each element h in the list the secret-key in the pre-image of h corresponds to one of the right eligible voters and if this voter signed a YES h is in the first list and if this voter signed a NO h is in the second list.

In this way, if a YES vote for C is counted in another proof the NO list in the second proof will have the same value $\mathsf{Hash}(\mathsf{Sk}_C, \mathsf{id})$ from the YES list in the first proof and this bad voter can be removed without disclosing his identity. After having removed all bad voters from all proofs, the smart contract can just take the union of all non-bad voters for YES and NO and tally them.

Note that the information submitted by each aggregator would grow linearly with the number of voters. Moreover, observe that the above approach does not contradict the impossibility result of aggregatable ZK proofs. Indeed, in the impossibility result, we are supposing the statements to consist of only the claimed result and so be of size independent of the number of voters who cast a vote. Instead, in the proposed mitigation the statement includes as auxiliary information a list of values that depend on all the secret-keys of voters who cast a vote.

3.3. Impossibility for succinct aggregatable non-interactive proofs

Consider the following generalization of the previous notion of aggregatable non-interactive proofs.

Definition 2. *A non-interactive proof system (K, P, V) is aggregatable if it is endowed with an efficient aggregator algorithm* Agg *that satisfies the following property.*

Let S be a set of voters. In the following let σ be the CRS output of K on the security parameter.

Let c_1 be the claim that there are $|S|$ YES signatures for voters in S and 0 NO signatures, let c_2 be the claim that there is a single YES signature and 0 NO signatures, and let c_3 be the claim that there are $|S| + 1$ YES signatures and 0 NO signatures.

Let π_1 be a proof computed by P with CRS σ, claim c_1 and witness consisting of YES signatures of all voters $i \in S$ computed with their respective secret-keys $\{\mathsf{Sk}_i\}_{i \in S}$.

For all $j \in [D]$, where D is the total number of eligible voters in the census, it holds the following.

Let π_2 be a proof computed by P with CRS σ, claim c_2 and witness consisting of a YES signature of voter j computed with its secret-key Sk_j.

Let π' be the output of $\mathsf{Agg}(\sigma, \pi_1, \pi_2)$.

Then the following properties hold.

- *Property F_1. If $j \in S$ then: $V(\sigma, c_1, \pi') = 1$ and $V(\sigma, c_3, \pi') = 0$.*
- *Property F_2. If $j \notin S$ then: $V(\sigma, c_3, \pi') = 1$.*

It is easy to show that a SNARK, satisfying the latter notion of aggregatability, might be used to guarantee Property 1 discussed in Section 2 in the obvious way, so it would solve the referendum problem. In the rest of this section we prove that no non-interactive proof system can be both aggregatable (in the sense of Def. 2) and succinct.

Kolmogorov Complexity. We now recall some basic facts about Kolmogorov Complexity.[10]

Definition 3. *Let n be an integer > 0. A string $w \in \{0,1\}^\star$ has Kolmogorov complexity $\geq n$ if there exists no computer program of length $< n$ that (without any input) outputs w. In symbols we write $KC(w) \geq n$.*

A string $w \in \{0,1\}^\star$ has conditional Kolmogorov complexity $\geq n$ given another string $y \in \{0,1\}^\star$ if there exists no computer program of length $< n$ that, on input y, outputs w. In symbols we write $KC(w|y) \geq n$.

The following lemma follows from the pigeon principle.

Lemma 1. *Let $n > 0$. The set $\{0,1\}^n$ contains at least one string w such that $KC(w) \geq n$. Moreover, this holds even conditioning on any other string $y \in \{0,1\}^\star$, that is the set $\{0,1\}^n$ contains at least one string w such that $KC(w|y) \geq n$.*

Theorem 2. *A non-interactive proof system that is aggregatable in the sense of Def. 2 cannot have proofs of length sublinear in the total number of eligible voters.*

Proof 2. *In the following we let s be the list of all pairs of public- and secret-keys of all voters. Observe that the census can be computed from s.*

For any string $x \in \{0,1\}^\star$, the hamming weight $\mathsf{hw}(x)$ is the number of bits of x that are equal to 1.

Given CRS σ, and s consider a string $w \in \{0,1\}^D$ such that $KC(w|\sigma, s) \geq D$. The existence of such string is guaranteed by Lemma 1.

For any $z \in \{0,1\}^n$, in the following we write $P(\sigma, (0, \mathsf{hw}(z)), z)$ to denote a proof computed for the claim that there are 0 NO signatures and $\mathsf{hw}(z)$ YES signatures computed with the YES signatures of voters in the set $\{i|z_i = 1\}$.

Consider the following program A that depends on constants $\mathsf{hw}(w)$, the total number of eligible voters D, proof $\pi_1 \leftarrow P(\sigma, (0, \mathsf{hw}(w), w)$, and takes as input CRS σ and string s defined above.

$A[\mathsf{hw}(w), m, \pi_1](\sigma, s)$ computes the following.

- *For all $j \in [D]$ set $e_j \in \{0,1\}^n$ to be the string that is 1 in position j and 0 elsewhere.*
- *Use s to get the set of all public- and secret-keys of all D eligible voters, and compute the Merkle root census C for them.*
- *Compute the YES signatures $\{\sigma_j\}_{j\in[D]}$ of all eligible voters.*
- *For j from 1 to n does the following.*
  - *Compute $\pi_2 \leftarrow P(\sigma, (0, 1), j)$.*
  - *Compute $\pi' \leftarrow \mathsf{Agg}(\sigma, \pi_1, \pi_2)$.*

- *Set* $z_j = 1$ *if and only if* $V(\sigma, (0, \mathsf{hw}(w)+1), \pi') = 0$.

Output z

By properties F_1, F_2 *of Def. 2,* A *outputs* $z = w$. *The length of* A *is bounded by the length of* π_1, *the length of* $\mathsf{hw}(w)$, *the constant* $2\log(D)$ *needed to perform the loops and another constant* c. *So,* $|A| \leq |\pi_1| + |\mathsf{hw}(w)| + 2\log(D) + c$, *that is, asymptotically,* $\leq |\pi_1| + 4\log(D)$ *that implies* $KC(w|\sigma, s) \leq |\pi_1| + 4\log(D)$.

By the fact that $KC(w|\sigma, s) \geq D$ *we have that* $|\pi_1| + 4\log(D) \geq D$ *and thus* $|\pi_1| \geq n - 4\log(D)$.

We conclude that there is no aggregatable non-interactive proof that has proofs of length sublinear in the total number of eligible voters in the census.

References

[1] Nouns DAO, Private Voting Research Sprint, https://prop.house/nouns/private-voting-research-sprint, 2023.

[2] Arnacube, R. B. Pau Escrich, A. Kampa, Ovote: Off-chain voting with on-chain trustless execution, https://github.com/aragonzkresearch/research/blob/main/drafts/ovote.pdf, 2023.

[3] R. C. Merkle, A digital signature based on a conventional encryption function, in: C. Pomerance (Ed.), Advances in Cryptology – CRYPTO'87, volume 293 of *Lecture Notes in Computer Science*, Springer, 1988, pp. 369–378.

[4] J. Groth, On the size of pairing-based non-interactive arguments, in: M. Fischlin, J. Coron (Eds.), Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II, volume 9666 of *Lecture Notes in Computer Science*, Springer, 2016, pp. 305–326.

[5] R. Gennaro, C. Gentry, B. Parno, M. Raykova, Quadratic span programs and succinct NIZKs without PCPs, in: T. Johansson, P. Q. Nguyen (Eds.), Advances in Cryptology – EUROCRYPT 2013, volume 7881 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 626–645. doi:10.1007/978-3-642-38348-9_37.

[6] Polygon Hermez, use recursion to build ZK EvM, https://docs.hermez.io/zkEVM/Overview/Overview/#hybrid-mode-for-on-chain-data-availability, 2023.

[7] Scroll, ZKEvM, https://scroll.io/blog/zkEVM, 2023.

[8] A. De Santis, S. Micali, G. Persiano, Non-interactive zero-knowledge proof systems, in: C. Pomerance (Ed.), Advances in Cryptology – CRYPTO'87, volume 293 of *Lecture Notes in Computer Science*, Springer, 1988, pp. 52–72.

[9] P. Ananth, A. Deshpande, Y. T. Kalai, A. Lysyanskaya, Fully homomorphic nizk and niwi proofs, Cryptology ePrint Archive, Paper 2019/732, 2019. URL: https://eprint.iacr.org/2019/732, https://eprint.iacr.org/2019/732.

[10] M. Li, P. Vitanyi, An introduction to Kolmogorov complexity and its applications, Springer Verlag, 1997.