The Referendum Problem in Anonymous Voting for Decentralized Autonomous Organizations

Artem Grigor, Vincenzo Iovino

Giuseppe Visconti, Univ. of Salerno

5th Distributed Ledger Technology Workshop (DLT 2023), May 25–26, 2023, Bologna, Italy

Voting for Decentalized Autonomous Organizations and Web3

- DAOs are members-owned communities without centralized leadership.

- One of the main DAOs' functionalities is to perform 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*

Shortcomings of web3 voting

- Most of the DAOs operate with contracts over the Ethereum network contracts.
- Voting can be done by requesting each member to send either standard digital signatures (no privacy) or, if privacy is a must, a SNARK proof of membership in a given *census*.
- Cryptographic operations consume a huge quantity of GAS.

Minimizing the GAS cost for Anonymous Referenda

- Recently several projects in the web3 space are working on anonymous voting.

- Anonymous voting is done via SNARK proofs whose verification costs onchain about 300-400k GAS per each voter's proof.
- A folklore solution to cut costs is the following:
  - Voters send their own digital signatures (if Aggregator is trusted for privacy) or SNARK proofs to an off-chain Aggregator.
  - The Aggregator can then compute a SNARK proof of e.g. knowledge of m signatures for YES and n sig. for NO and sends it to the smart contract.

The 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?

The naïve solution does not work

- Consider solutions in which there is only a single authorized Aggregator that can communicate with the smart contract.
- System P1: there is a single Aggregator authorized to submit results and proofs on-chain (e.g., the smart contract accepts inputs only signed by the given Aggregator).
- Trivial system P2: the authorized aggregator submits to the smart contract the result *in the clear* (zero cryptography).
- P1 is no better than P2: in P1 the aggregator can still receive 10 signatures for YES and 11 for NO and sends to the smart contract a proof for e.g. 10 signatures for YES and 9 for NO.
- Summing up: assuming that there is a single Aggregator authorized to communicate with the smart contract → the aggregator is completely trusted → P1 is not better than P2.

Setting and Capability of the attacker

- Since the smart contract cannot accept data from a single authenticated node, we must assume there are multiple nodes over the network that can send data to the contract.
- The attacker is capable of corrupting a subset of voters.
- 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 contract.
- 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.

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 policy is *good* if it satisfies a *Property.*
- Example of property: property P_1.
  - Consider the set SY (SN) of voters who cast a vote for YES (NO) to some node (any computer in the world). Then SY (resp. SN) includes v iff a voter v sent a YES (resp. NO) to some (possibly dishonest) node. Let Bad = $SY \cap SN$, that is the set of bad voters who cast a vote both for YES and NO, and let us denote by SY' = SY − Bad and SN' = SN − Bad the new sets in which we remove the bad voters.
  - Then, the property P_1 is formalized as follows:
    - A policy is good if the decision output by the policy is equal to YES if $|SY'| > |SN'|$, is equal to NO if $|SN'| > |SY'|$ and is equal to TIE if $|SY'| = |SN'|$.

Attempt to address the Referendum Problem

- Consider the following policy:
  - From the many accepted proofs consider the one with the highest number of votes and outputs the result corresponding to that proof
  - So, for example if the contract receives a proof π_1 for result (10,12) and a proof π_2 for result (13,12), the second one has the highest number of total votes (25) and the contract should output as result (13,12).

Real-World Counter-Example (1)

- Consider the following real-world situation in which the contracts receives the following:
  - Proof π_1 for (3, 2) from some node 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)
- We also assume that N_1 has been malicious in removing NO signatures of (B, J, S) that N_1 received. The other nodes acted honestly based on the signatures they received.
- According to the policy, 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.

Real-world Counter-Example (2)

- 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.
  - 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, 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, according to property P_1 the actual result should be NO but the contract, following the policy outputs YES, contradiction!

Another attempt to address the Referendum Problem

- Maybe the previous property was too stringent?
- Consider the following alternative property P_2:
  - We can assume that there is a single “trusted” node TN in the sense that honest voters are prescribed to send their signatures to it.
    - Note: this assumption may apply not just in the context of this property.
  - 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 received m signatures for YES and n for NO and $m > n$ the result is YES, etc.

Analysis of Property P_2

✓ Property P_2 is very close to traditional e-voting in the sense that it is implicitly stating that there is a single source of truth: whatever is published on the trusted node TN is the «truth» and the smart contract should give results that are consistent with just TN and nothing else. So the property makes a lot of sense!

✗ Unluckily the property is unachievable as well:

- The idea is that if it were achievable with respect to previous policy, the contract could discriminate which data are coming from TN even without any authentication. See paper for formal counter-examples.
- In the counter-example we strongly use the fact that an adversary can perform copy-and-paste attacks: that is uncorrupted voters are honest in sending their data to just TN but the attacker can replicate such data on other nodes to "*confuse*" the contract.
- This property is not just unachievable with respect to the previous policy but it seems very hard to come up with other policies that achieve P_2.

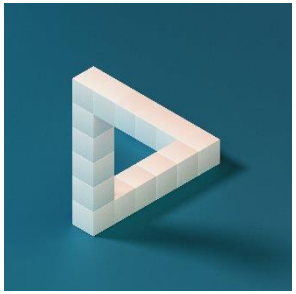
Achievability of Property P_1

- Are there other policies that achieve property P_1?
- 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 "*aggregated*" 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 resulting proof preserves *privacy* and *succinctness*.

Aggregation would achieve Property P_1

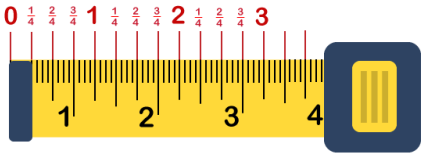
- This ideal cryptographic primitive can be seen as a solution to the referendum problem while preserving privacy and succinctness: technically it would be a *good policy* for property P_1.
- Indeed, consider the previous counter-example against P1:
  - Suppose that N_2 is the node to which honest voters are prescribed to send their votes.
  - From the 3 proofs π_1, π_2, π_3, anyone can generate a single proof π of the result (3, 4) corresponding to YES signatures of (A, V, R) and NO signatures of (B, J, S, M) where the signatures of C have not been counted because C voted both YES and NO.
  - Then, notice that a policy that uses these aggregatable SNARKs to select the result would output as result NO as prescribed by P_1.

Impossibility results for non-interactive aggregatable ZK proofs



- Theorem 1. There is no aggregatable NIZK proof system.
  - Idea: let π_1 be a proof for claim (2,0) computed with YES sig. of (A,B), π_2 be for claim (1,0) computed with sig. of (C), and π_3 be for claim (1,0) computed with sig. of (A). Aggregation implies that from π_1, π_2 one can compute an aggregated proof for the claim (3,0), and from π_1, π_3 one can compute an aggregated proof for the claim (2,0), so adversary can distinguish which one among π_1 and π_2 was computed with sig. of A rather than sig. of C.
  - See paper for details. Theorem 1 extends to witness indistinguishability as well.

Impossibility results for non-interactive aggregatable short proofs



- Theorem 2. There is no aggregatable succinct (short) argument system.
  - Proof idea:
    - Kolmogorov complexity tells us that in a set of 2^n strings there is at least one string z that is uncompressible in the sense that no computer program of length less than n can print z. This also applies to programs that have hard-cored another string y.
    - Then one can consider a proof π for the claim that there are 0 NO sig. and a YES signature for voter i iff z_i=1.
    - A program P can use proof π that is, by hypothesis succinct, and the aggregation property to discriminate whether voter i voted YES and so can derive all bits of z and prints out z.
    - Overall, we construct a program P whose size depends on a short proof π (and other auxiliary information) that prints a string z of high Kolmogorov complexity (i.e., uncompressible), a contradiction.
  - See paper for details.

What our results do not cover

- We showed the hardness of facing the referendum problem.
- However, observe that our counter-examples and impossibility do not contradict the feasibility of the following solution:
  - A voter submits as ballot his own YES/NO preference + related SNARK proof of membership.
  - The smart contract hashes all voter’s ballots so as to obtain in the end a digest H that compresses all of them.
  - An aggregator submits to the contract a proof that the digest H is such that its preimage contains m (resp. n) votes for YES (resp. NO) along with corresponding accepted proofs.
  - This approach would require recursive proofs and technically is not succinct because it requires anyway to store an amount of data that is proportional to the number of voters.

Questions?