Private Dispute Resolution on Ethereum

Private Dispute Resolution on Ethereum

Andrea Gangemi
Department of Mathematics Sciences
Politecnico di Torino
Turin, Italy
andrea.gangemi@polito.it

Aida Manzano Kharman
Department of Design Engineering
Imperial College London
London, United Kingdom
aida.manzano-kharman17@imperial.ac.uk

***Abstract*—We present a dispute resolution protocol that can be built on the Ethereum blockchain. Unlike existing applications, it is private by design through the use of zero-knowledge proofs. The protocol is resilient to Sybil attacks and provides increased collusion resistance amongst participating parties. A resolution to the dispute is guaranteed, whilst ensuring the users have the final say on the chosen resolution. The proposed protocol rewards stakeholders through a social incentive mechanism leveraging Soulbound tokens, and rewards agents who behave honestly, as opposed to favouring the wealthy. To our knowledge, this is one of the first dispute resolution protocols to implement governance through reputation as opposed to token-based voting. Furthermore, it is fully viable, given that all its necessary components are currently operating on the Ethereum blockchain.**

***Index Terms*—Dispute Resolution, Ethereum, Quadratic Voting, Soulbound Tokens, Zero Knowledge Proof**

I. Introduction

Users must have a means to raise a dispute in Web3 applications. If the application is not providing the service as intended, consequences can be dire for the users. For instance: Binance [4] must ensure that all tradeable tokens on their site are related to valid Blockchain projects, and the Proof of Humanity protocol [26] must only add real humans to its registry.

Blockchain dispute resolution mechanisms: Conflict resolution in Distributed Ledgers Technologies (DLTs) and decentralised applications (dApps) is not yet regulated by law. The dApp space is rife with disputes, some leading to significant financial damage to their participants[1]. There is no dedicated protocol to protect victims or hold wrong-doers accountable. As a result, a number of dispute resolution projects have emerged as an attempt to address this issue. Kleros is the most notable one to date [16].

However, all these dispute resolution services are based on *arbitration*, a form of *Alternative Dispute Resolution* (ADR). The limitation of this procedure is that the conflict passes into the hands of third parties, arbitrators, who ultimately decide on the resolution that the users must accept. DLTs, however, were designed to avoid relying on third parties for decision-making, as this would re-introduce centralisation. Hence, we consider solutions following this approach unsuitable. *Mediation* is another *Alternative Dispute Resolution* method, which can be considered an improvement over current techniques. In it, the users defend their stance in a conflict to one or more mediators, who eventually propose a mediation agreement. The users can choose to reject the agreement if they do not find it satisfactory. Third parties involved in the mediation process can therefore propose, but not impose. The drawback is that if the users reject the agreement, the dispute is not resolved and they must resort to another process.

*a) **Our contribution**:* We present a new proposal for dispute resolutions on the Ethereum blockchain, given that Ethereum is most frequently used as a service layer for dApps due to its support of smart contracts[2]. Our protocol is private by design, and makes collusion between judges[3] difficult. It also allows users to have the final say on the accepted resolution. We note that in future work this proposal may be extended to use cases beyond Ethereum.

*b) **Structure of the paper**:* The paper is organised as follows: in section II we briefly recall the state-of-the-art of dispute resolution mechanisms. Section III outlines the functioning of the key working components of the dispute resolution mechanism proposed. Section IV introduces the framework upon which our protocol is based. Later, section V describes the social and financial incentives of the system, section VI explores possible attacks and how our protocol is able to resist them, and section VII summarizes how our framework can be implemented. Finally, section VIII outlines paths for possible future work.

II. Related Work

A. Dispute Resolution Mechanisms

We organise the survey of existing dispute resolution mechanisms according to the following taxonomy:

- **Global mechanisms:** they are Decentralised Autonomous Organisations (DAOs) that can be integrated into other DAOs with the purpose to solve disputes arising from any other dApp in the ecosystem. Kleros [16] is one such example.
- **Local mechanisms:** these include dispute resolution mechanisms native to a given DAO, where the purpose of the DAO is not exclusively to enable a dispute resolution,

[1] As an example, see the following website: DAO Coup, Vice.

[2] A smart contract is a program that will automatically execute a code once certain conditions are met. It does not require intermediaries and allows for the automation of certain tasks [5][34].

[3] In our protocol, the judges are the agents tasked with assessing the conflict between two or more parties in the dispute and proposing a solution.

rather, it requires one for its own functioning. Aragon [2] is an example of a DAO with a built-in dispute resolution mechanism.
- **External mechanisms:** they are traditional dispute resolution mechanisms (arbitration, mediation, and so on) that operate entirely in an off-chain court, and then notarise the result on-chain.
- **Enhancing mechanisms:** there are also some applications that enhance a smart contract such that it becomes legally binding [25].

We focus on the first category, since our proposal is designed to be a global mechanism that is compatible with any Ethereum DAO, and also describe how it compares to other notable dispute resolution mechanisms such as Aragon and Kleros for context.

Kleros [16, 21] is the most active dispute resolution platform on the Ethereum blockchain. Its service is active since 2019 and it has solved more than 1500 disputes to date. Judges are randomly selected and then asked to vote amongst different resolution options using *plurality voting*. If users are dissatisfied with the outcome of the dispute, they can appeal and the process starts again with a bigger number of arbitrators. Examples of disputes that have been effectively resolved by Kleros include the following areas: curated lists, escrow, insurance, token listings, and some minor areas such as social networks or Gitcoin grants [17–19].

Aragon [2, 8] is a DAO that enables the development and maintenance of decentralised organisations running on the Ethereum Virtual Machine. Aragon token holders can access the Aragon DAO services, and one of these services is the Aragon Court, which solves disputes arising from the Aragon DAOs through a crowd-sourcing method which works exactly like the Kleros one. The Aragon Court has solved less than 50 disputes to date.

They propose using three different arbitration protocols for dispute resolution, depending on the level of severity of the conflict in question.

In the past years, many other blockchain dispute resolution projects failed or were never implemented. Examples include Sagewise, Oath, Juris (for all of them, see [24]) and Aspera [3, 27]. Aspera was one of the first ideas designed to provide a dispute resolution service through mediation. However, the project failed partly because of the complexity of the design, which was largely based on machine learning and artificial intelligence, and partly because mediation does not yet seem to be necessary on the blockchain, as is confirmed by the success of Kleros, in which most disputes are handled through the use of a simpler arbitration service.

B. Voting

In our proposal, we require the use of a voting protocol, firstly for judges to vote for which party they believe is in the right, and secondly for each party involved to vote on their desired dispute resolution outcome. We consider two components of voting: the tallying algorithm (also known as the voting scheme or protocol) and the voting *platform*, where the voting occurs. Research on tallying algorithms used in dApps is still young. The work in [10] surveys the most commonly used voting schemes in DAOs. The most commonly used are permissioned majority voting, token-based majority voting, conviction voting and quadratic voting. The first two are the simplest schemes, which means they are easiest to implement in smart contracts, thus enabling on-chain voting. The latter two aim to prevent easy acquisition of power by the most wealthy participants. Conviction voting grants more voting power to voters who do not change their opinion and quadratic voting makes acquiring more votes quadratically more expensive[20]. The authors in [10] and [12] propose a set *criteria* to evaluate the suitability of voting schemes. We consider them to be incomplete, elaborate on them and use them to select the most suitable tallying algorithm and voting platform for our application. [12] does not consider *fairness* in their matrix to evaluate DAO voting mechanisms, and their *security* notion is a heuristic with no formal mathematical or computational formalisation. The *fairness* definition in [10] is only applicable if the voting protocol is a *One-Person-One-Vote* (1P1V) scheme. A 1P1V scheme is used in democratic elections, assuming a functioning Sybil-protection mechanism. Given that this is harder to achieve in the context of dApps, whilst we advocate for 1P1V schemes, we also consider the fairness and security of voting schemes when voters may purchase more voting power (i.e: a *One-Dollar-One-Vote* mechanism). The latter is akin to shareholder ownership models, and for some dApps it may make more sense as a solution.

In terms of *security*, the voting scheme must provide privacy guarantees. We consider the following notions of privacy: a voter has the right to secret ballots and should not be able to prove how they voted to anyone. This notion is formalised in the definition of *Ballot Secrecy*, as defined in [33]. Without it, voters may be coerced, sell their votes, DAOs to buy votes (vote buying cartels) can arise [9] where smart contracts can be written to execute and automate transactions of votes for money. Finally, the voting scheme must ensure *verifiability*. Voters should be able to verify that the election outcome does indeed represent the voters' votes. This ensures legitimacy of the election outcome.

We propose considering the following criteria:

Fairness: the voting mechanism does not disproportionately favour the wealthier voters.

Speed: the voting mechanism can return an outcome in a timely manner.

Privacy: voter's votes are secret, and they cannot prove how they voted to a third party. The voting scheme used satisfies *Ballot Secrecy*, as defined in [33].

Verifiability: voters must be able to verify the election outcome indeed reflects their votes.

We conclude that quadratic voting is the most suitable option for our application. This scheme makes it expensive to mount a heist, but it will favour malicious agents that are very wealthy. It is also known to be vulnerable to Sybil attacks.

Neither of these are a relevant concern in the context of our proposal, because voters have a limited amount of wealth they can spend on votes[4], and cannot register new identities once the dispute resolution process has been initiated. We elaborate further in Section IV. Quadratic Voting is faster to return an outcome than conviction voting[5], and fairer than the straightforward majority voting and its token-based equivalent: token-based quorum voting, since it makes monopolisation of voting power prohibitively expensive. Given this compromise, we select this voting mechanism for our proposal. It is fair, faster than existing alternatives and privacy and verifiability are guaranteed through its implementation in the MACI voting platform. We elaborate on the implementation and its security guarantees in section III-A0b, and further justify the choice of this scheme in section III-C.

III. Preliminaries

In this section, we summarize the privacy protocols *Semaphore* and *MACI* (Minimal Anti-Collusion Infrastructure) that will be used in Section IV. These protocols use *zero knowledge Succinct Non-interactive ARgument of Knowledge*, (*zk-SNARKs*). Zk-SNARKs allow a user to prove possession of some information, without revealing said information. Namely, zk-SNARKs are characterised by the following: the proof is succinct (meaning the proof is smaller than the statement being proven) and the proof is fast to verify for the verifier. On the other hand, the prover time grows (approximately) linearly with respect to the number of operations they have to perform. Semaphore and MACI both use the Groth16 zk-SNARK [14], which has prover complexity $O(n \log_2(n))$ (where n is the number of operations), while the proof size and the verification time are constant. When used together with a blockchain, the succinct proof, π, is created off-chain, and then sent to a smart contract that rapidly verifies its correctness.
Then, we outline the functioning of *Proof of Humanity* and how it can be integrated with Semaphore. Subsequently, we introduce *quadratic voting*, and further justify its use over other existing alternatives. Finally, we summarise how *Soulbound tokens* are used in the proposed mechanism.

A. Zero knowledge protocols

Ethereum currently has more than twenty open projects that use zero-knowledge techniques to enhance the privacy or the scalability of underlying protocols [28]. Two of these projects, Semaphore and MACI, are useful building blocks for the idea proposed in this paper. We remark that both protocols are already existing and used by various projects in the Ethereum ecosystem.

a) ***Semaphore****:* [31] is a zero-knowledge protocol which allows Ethereum users to prove their membership in a group and send signals such as votes or endorsements without revealing their identity. Namely, Semaphore provides three functionalities:

- *Creation of private identities*. A user that joins a Semaphore group receives a secret/public key pair $(\mathsf{sk}, \mathsf{pk})$. The secret key is a tuple of three values $\mathsf{sk} = (\mathsf{IdTrapdoor}, \mathsf{IdNullifier}, \mathsf{IdSecret})$, where IdTrapdoor and IdNullifier are generated randomly, while $\mathsf{IdSecret} = H(\mathsf{IdTrapdoor} \,||\, \mathsf{IdNullifier})$ and H is a hash function. The nullifier is necessary to avoid users signaling more than once. The public key is the hash of IdSecret where IdSecret: $\mathsf{pk} = H(\mathsf{IdSecret})$. The private key sk is used to generate zero-knowledge proofs;
- *Insertion of an identity into a group*. To be part of the same group, all the users must share a common trait. In the context of our application, this trait can be proof of a judge's area of expertise relevant to the dispute. Everyone is then sure that all the members possess this trait, but they do not know the real identity of these members;
- *Sending of anonymous signals*. A *signal* is a signed message which is broadcast on-chain. A signal contains the following data:
  - A vote.
  - A membership proof showing that the user is a member of a Semaphore group.
  - A proof that the same user created both the signal and the first membership proof.

  For most applications it is mandatory that every member can signal only once. For this reason each signal also contains two additional values: a public one, ExtNullifier, which is usually the ID of the Semaphore group, and then the digest $\mathsf{Nullifier} = H(\mathsf{IdNullifier} \,||\, \mathsf{ExtNullifier})$. Since IdNullifier is part of the private key of a user, if two different signals have the same value Nullifier it means that the same user has signaled twice. To summarise:

$$\mathsf{Signal} = (\mathsf{Data}, \mathsf{Proof}(\mathsf{pk}),$$
$$\mathsf{Proof}(\mathsf{Data}, \mathsf{Proof}(\mathsf{pk})), \mathsf{ExtNullifier}, \mathsf{Nullifier}).$$

Semaphore can thus be regarded as a Sybil-protection mechanism: each signal sent contains certain zero-knowledge proofs, generated off-chain and validated on-chain, about the sender's membership of a certain group, as well as the validity of the signal itself. More details about the implementation of the Semaphore circuits or their smart contracts are available on the Semaphore website [31].

b) ***MACI****:* stands for *Minimal Anti-Collusion Infrastructure* and it is a protocol that allows users to vote on-chain with increased collusion resistance. It was proposed in [35]. All transactions on a blockchain are public, so in on-chain voting platforms a voter can easily show to a briber which option they voted for. MACI counters this issue by allowing each voter to encrypt their vote. Each voter shares a key with the trusted coordinator, who is tasked with decrypting and tallying the votes off-chain. The coordinator then uses zk-SNARKs to prove that the tally has been correctly computed, using only valid votes, without revealing the vote of each user.

[4]These are assigned to them *a priori* by the judges in the form of *voice credits.*

[5]In conviction voting, user's voting power increases the longer they leave their vote unchanged.

Before voting, each user, who must already possess a secret/public key pair $(\mathsf{sk}, \mathsf{pk})$ (which can be generated when joining a Semaphore group), registers their public key pk in a smart contract SC_1. Each registered user obtains some *voice credits*, which are the number of votes a user can cast when voting on a dispute resolution. They can vote with any address, but the transaction continaing their vote must include the registered public key. Finally, each user I shares also a (symmetric) key $\mathsf{SharedKey}_{I,C}$ with the trusted coordinator C, which is used to encrypt and decrypt transactions. To vote, the user I will send an encrypted transaction to some poll smart contract SC_2, containing the following data:

$$\mathsf{Transaction} = \mathsf{Enc}_{\mathsf{SharedKey}_{I,C}}(\mathsf{Sig}, \mathsf{Command}),$$

$$\mathsf{Command} = (\mathsf{pk}_I, \mathsf{Vote}_{\text{option}}, \mathsf{Vote}_{\text{amount}}).$$

Sig represents the signature of the user that is sending the transaction (which is obtained using the secret key sk_I), while $\mathsf{Vote}_{\text{option}}$ is the list of projects that the user wants to vote for. Finally, $\mathsf{Vote}_{\text{amount}}$ is the list containing the amount of voice credits the user has allocated to each project they have decided to support.

Users can override their previous vote if they sign a new transaction with their secret key sk_I. In this case, the coordinator will consider only the last message as valid.

Users can also override their public key, if they sign a new transaction that contains in it a different public key $\widetilde{\mathsf{pk}_I}$ whilst still using their secret key sk_I to sign. That is

$$\mathsf{Transaction} = \mathsf{Enc}_{\mathsf{SharedKey}_{I,C}}(\mathsf{Sig}, \mathsf{Command}),$$

$$\mathsf{Command} = (\widetilde{\mathsf{pk}_I}, \mathsf{Vote}_{\text{option}}, \mathsf{Vote}_{\text{amount}}).$$

From then on for a transaction to be considered valid it must contain the public key $\widetilde{\mathsf{pk}_I}$. This feature is known as *public key switching*. After this moment, transactions must be signed with the secret key $\widetilde{\mathsf{sk}_I}$. Public key switching can be used to avoid bribes, since no one except the user and the trusted coordinator knows if the transaction sent will be considered valid after the decryption.

After the voting period, the coordinator will use a third smart contract SC_3 to collect valid votes only. Then, it performs off-chain the tally of the votes and publishes the results. During this process, the coordinator creates two different zk-SNARKs proofs:

- the first proof is published to prove that SC_3 contains only the valid votes, without revealing their content;
- the second proof is created to show that the tally of the votes was done using only valid messages, and that the tally was correctly computed.

Both proofs are verified by another smart contract SC_4, specifically built to read MACI proofs. The commitment of the tally can be published on-chain. As default, MACI works with quadratic voting, but more traditional voting systems can be selected as well.

MACI relies on the trust assumption that the coordinator is honest. If that is not the case, the security of the protocol is compromised, as the coordinator may reveal voter's votes. The protocol does not prevent a coordinator from being malicious, however, the trusted setup for the Groth16 zk-SNARKs that MACI uses may be computed using a multi-party trusted setup. This increases the confidence that the coordinator may not output false tallying proofs. We select MACI because it is the only current on-chain voting platform that does not use public votes, or decrypt votes publicly after the voting period ends. Whilst we note that MACI does not satisfy formal notions of Ballot Secrecy [33], no current on-chain voting scheme has been proven to do so, and that delivering an on-chain voting platform that does is an open research question beyond the scope of this work. It also provides verifiability, as the coordinator must provide tallying proofs and proofs that only valid votes were tallied. We require an on-chain platform to ensure the results are binding.

B. Proof of Humanity

[26] (PoH) is a decentralised proof-of-personhood solution. It ensures that every registered account is owned by a real person and that every user holds one account. Joining the protocol is straightforward: an interested person uploads a video of themselves, and then, to be approved, an already approved user needs to verify them. A period of time must elapse during which that user can be challenged: this can happen if the user that is trying to register is not considered human, or if it already has an account.

As shown in the document [39], Proof of Humanity can be integrated together with Semaphore to solve certain privacy problems. The project, called *Zero Knowledge Proof of Humanity* (zkPoH), consists of a smart contract that only allows one to register as a member of a Semaphore group if the subscriber is already registered in PoH. In this way, we are sure that each member of the Semaphore group is a real person, and they can issue signals without revealing their identity.

C. Quadratic voting

As we have seen in Section II, *quadratic voting* [20] is an alternative to other more classic voting modes, like *One-Dollar-One-Vote* (1D1V), where each user can vote as many times as they want and each vote costs one dollar, or *One-Person-One-Vote* (1P1V), where each user can vote exactly once. In this voting model, every person can vote as many times as they want, but voting n times will cost them n^2.

In certain situations, it is beneficial to enable *negative voting*, i.e. a user's vote towards a project is not intended to contribute towards that project, but to penalize it in relation to all others. Negative voting has been proposed by Vitalik Buterin in the context of quadratic voting [36].

As the next proposition shows, quadratic voting is a better solution compared to 1D1V, when each voter has a different number of votes allocated.

Proposition 1: Suppose that two users A and B have a different quantity of votes, V_A and V_B, to allocate to different proposals. Then, if negative voting is possible, one-dollar-one-vote mechanism has an always winning strategy for the user

with the higher amount of votes, while this strategy does not exist in the case of quadratic voting, when
$|V_B - V_A - 2y_2| < 2\sqrt{V_A y_2}$ with $y_2 \leq V_B$.

Proof 1: Without loss of generality, suppose that $V_A > V_B$.

- **1D1V**: in the case of *one-dollar-one-vote*, a winning strategy for the user A is to simply go all-in to the proposal they prefer, suppose p_1. In fact, user B can either go all in to another proposal, say p_2, or try to use negative voting on the proposal p_1 and then use their remaining votes, say y_2, for the proposal p_2. In the former, clearly p_1 is the winning proposal since $V_A > V_B$; in the latter, user B wants that

$$V_A - y_1 < y_2, \qquad y_1 + y_2 = V_B.$$

  However, proposal p_2 is the winning one if and only if $V_A < y_1 + y_2 = V_B$, which is impossible by hypothesis.
- **Quadratic voting**: suppose again that A goes all-in on the proposal p_1. Clearly, if B goes all-in on another proposal p_2, they will never win since $V_A > V_B$. However, by using negative voting on the proposal p_1, they can prevent A from having a winning strategy under some circumstances. In this case, B wants that

$$(\sqrt{V_A} - \sqrt{y_1}) < \sqrt{y_2}, \qquad y_1 + y_2 = V_B.$$

  Re-arranging, the above equation becomes

$$V_A^2 - 2V_A V_B + (V_B - 2y_2)^2 < 0,$$

  and it can be shown that this holds in the range

$$V_A + 2y_2 - 2\sqrt{V_A y_2} < V_B < V_A + 2y_2 + 2\sqrt{V_A y_2}.$$

  Hence, unless $|V_B - V_A - 2y_2| > 2\sqrt{V_A y_2}$, user B is not guaranteed to lose every time if A plays a simple all-in strategy.

D. Soulbound tokens (SBTs)

These were introduced by Weyl, Ohlhaver and Buterin in 2022 [38]. They are non-transferable (but revocable), non-fungible and publicly visible tokens that encode subjective qualities like the reputation of a user or the authenticity of a piece of art. SBTs are held in wallets, and they are public by default. Nonetheless, varying degrees of privacy can be achieved leveraging cryptographic protocols such as zero-knowledge proofs.
Soulbound tokens can be granted to users by other users, dApps or DAOs. A user may thus possess a digital identity linked to the real one, which is represented by a list of SBTs, each of which provides different information about that person. Since these tokens are non-transferable, when a user does not follow the rules of a certain protocol they might receive one, thus showing the rest of the network their negative behavior. This type of token is therefore a useful tool to introduce social compliance into the blockchain world.

IV. The new Dispute Resolution Protocol

In this section, we describe in detail our proposal for a novel dispute resolution mechanism. In particular, the protocol aims to:

1) allow users to have the final say on the resolution of the conflict;
2) allow potential judges to participate without requiring them to stake tokens to vote on a dispute resolution;
3) prevent users and judges from changing their opinions after a certain time and ensure collusion resistance for the judges;
4) assign governance of the dApp to those who have built their reputation over time, resolving disputes and contributing to the development of the platform.

The dispute resolution process can be divided into two phases (plus one subscription phase):

1) *Phase 0*: users that are interested in judging register in a Semaphore group upon successfully earning a Proof of Humanity;
2) *Phase 1*: once judges are notified of a dispute, they will send a transaction to a certain smart contract, containing a vote in favour of a party in the dispute, and a possible solution to the dispute. At the end of this process, the MACI coordinator computes the tally of the votes and gives a score to each user involved in the dispute. Thanks to MACI's privacy techniques, users cannot know the individual scores assigned to them by each judge;
3) *Phase 2*: the users will be able to vote for their preferred resolutions to the dispute. The votes they have received during the first phase are equal to the *voice credits* each user is granted. These *voice credits* are the number of votes each user can cast in favour of a dispute resolution proposal. The proposal that receives the most voice credits will be enforced.

A. Phase 0: judges' registration

To participate, judges must be registered on both Proof of Humanity and Semaphore. This guarantees that each user has a real identity. If the registration is successful, each judge i will receive a secret/public key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$. This public key is then also registered on the MACI smart contract SC_1: this step is essential, as the MACI protocol will ensure the judges do not collude. This last step also assigns to each judge one voice credit, which will be used to score the users involved in the dispute.

B. Phase 1: voting and proposals by the judges

Suppose a dispute involving n different users arises. For simplicity, from now on, we assume $n = 2$, but the model can be easily generalized for $n > 2$. One of these users can activate the dispute resolution mechanism by sending to the smart contract a transaction containing an ETH fee f.
The other party will have to send a transaction with the same fee f to join the dispute: refusing to join makes the initiating

party the winner automatically. Note that there is no incentive to start a dispute fraudulently, as in this case the other party can simply join the conflict and the judges will choose them as the winner, penalising the misbehaviour of the party that started the conflict.
If both sides join the dispute, they must provide evidence to support their case. Since storing documents on the blockchain is an expensive operation, one solution can be to store data off-chain, saving within the blockchain only the hash of this data. Storing data off-chain is a secondary issue, which can be left to the users involved: one can choose a centralised solution, such as a cloud server, or a decentralised solution such as the *InterPlanetary File System* (IPFS) protocol [15]. Another alternative is the mechanism introduced by Kleros and explained in the ERC-1497 proposal [11].
There is also the need to set two time thresholds t_1, t_2: judges that want to solve the dispute need to start participating before time t_1, and they can express their opinion until time t_2, which represents the end of this first phase. We also suppose that a minimum number of judges M must vote, to guarantee a level of decentralisation. We note M must be odd if there are two disputing parties.
Every judge will vote using their voice credit, using a *One-Person-One-Vote* mechanism.

When the time t_2 has elapsed, the MACI coordinator computes the tally of the votes: the result is saved on-chain via a commit of the tally. The users will then be able to see the total scores V_A and V_B they obtained, but they will not know the individual votes they received from each judge.
At the end of the first phase, there are three options:

- $V_A = V_B$: both users received the same amount of total votes, that is no one received an advantage for the second phase;
- $V_A > V_B$: the user A received more votes than the user B. This does not mean that A won the dispute, but only that the judges expressed a preference toward that user;
- $V_A < V_B$: this case is symmetrical to the previous one.

C. Phase 2: users vote on the judges' proposals

Having reached this point, the users involved in the conflict will have the possibility to read all the m proposals made by the judges and vote on the ones they prefer. The voting mechanism used in this case is the quadratic voting, and the voice credits they can spend are equal to the scores V_A, V_B they obtained during Phase 1. For example, user A may have received 15 voice credits, while user B only has 10. In this case, to avoid always-winning strategies, users can also assign a *negative vote* towards the proposals they do not find appealing. This phase takes place off-chain, after the two parties sent a transaction with a commitment on their vote. There are a number of off-chain e-voting platforms available for this step. Existing solutions offer secret ballots, end-to-end verifiability [1], and increased coercion-resistance [6].
At the end of the process, the proposal that received the highest score is enforced.

Figure 1 depicts the overview of the proposed solution. In summary, the protocol operates as follows:
1. Both conflicting parties raise a dispute to the Dispute Resolution Application (DRApp).
2. Judges that participate to the DRApp (which are part of both PoH and Semaphore) can decide if they want to rule the dispute. These vote in favour of the party they believe is right and provide a proposal for the resolution.
3. The MACI coordinator tallies the votes, which are assigned to each party as voice credits, and outputs the list of proposals. The tally happens off-chain, and at the end of it the coordinator sends to the DRApp a transaction containing a commitment to this tally.
4. Each party votes on a proposal using their voice credits.
5. The winning proposal is implemented in the DAO. The judges receive their rewards for their work through the DRApp (see Section V).

V. Incentives

Judges and users are incentivised to use this system in two different ways: through an economic incentive and a social incentive.
Social incentives reward those who behave honestly and enable the functioning of the platform, whilst penalizing bad behavior. This is possible through the use of Soulbound tokens (SBTs).
Following Buterin's paper [38], the ultimate goal is to entrust the DAO governance to judges who enable the proper functioning of the platform, instead of to those who own more ERC-20 tokens, as is usually the case.

A. Incentives for users

For users who face a conflict, their main incentive is the resolution of the dispute. Furthermore, SBTs are issued to those users who have been cooperative during the dispute process and have complied with the agreement made. They are linked to the wallet of these users, so the entire Ethereum network can see how they behaved.
Similarly, SBTs are issued to those users who do not comply with the agreement made. For example, suppose the dispute is about removing a token from the pool of those that can be purchased on Binance. The final proposal accepted by the users is to remove this token from the platform within a certain period of time t. If after time t has elapsed, the token is still purchasable, an SBT will be associated with the wallet of the user who was in charge of removing it. This SBT has a negative meaning, because the entire Ethereum blockchain will see that the user managing that given wallet has not fulfilled the agreement made.
These SBTs can be used in the context of the DAO where the dispute happened, for example to show to the rest of the participants that they care about the development of the DAO itself. They can also be used for DAO internal voting. Users

[6]In order of appearance: Icons made by Freepik, Vitaly Gorbachev, LAFS, Freepik, monkik, Freepik from www.flaticon.com

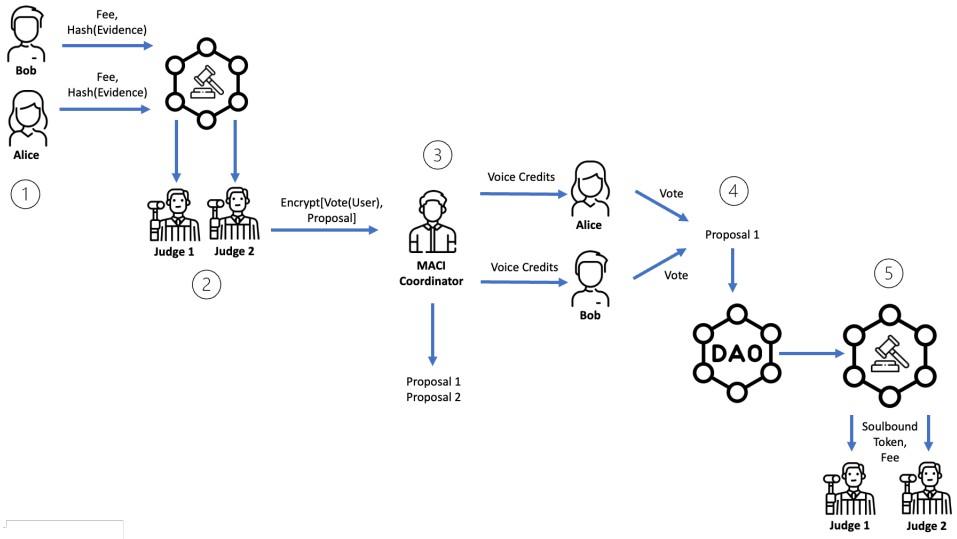

Fig. 1. Dispute Resolution Mechanism. Credit for icons given in:[6]

may bypass their negative SBTs by creating a new wallet. To overcome this issue, a solution could be to force them to subscribe to PoH before starting the dispute resolution. In this way, all participants will know of their new wallet.

B. Incentives for judges

Unlike Kleros, judges do not need a token stake to participate to a dispute. However, we still need to incentivise good behavior and especially to penalize bad behavior, otherwise a judge can simply register and then vote randomly each time, effectively offering a disservice to the users. For these reasons, we need both an economic incentive and a social incentive.

The economic incentive is the fee f required by the users to start the process: it will be distributed to the judge that suggested the winning proposal. All the other judges will not gain or lose anything (if we exclude the gas fee they have to pay in order to send their encrypted vote during the first phase of the protocol).

That is why a social incentive based on SBTs is necessary. Nevertheless, we must find a way to reward or penalize judges based on objective rather than subjective evaluations. For example, a judge who expresses a preference for the user who received fewer votes at the end of the tally has not necessarily voted in bad faith or carelessly, and for that reason they should not be penalized.

However, judges should be rewarded or penalized based on the quality of dispute resolution proposals they make. During Phase 2, users vote on the proposals received using the quadratic voting mechanism. They will give a positive score if they like the proposal received, but at the same time they can also give a negative score if the proposal does not satisfy them. The score obtained from the proposals will be converted into a *reputation score*, and assigned to the judges who made that proposal.

Next, define two thresholds $\epsilon_- < 0 < \epsilon_+$. If a judge has a reputation score lower than ϵ_-, they will be removed by the Semaphore group, hence they will not be able to express their opinion on the disputes anymore. Moreover, that judge will receive a SBT that certifies the bad behaviour towards the application. Notice that they cannot create a new account and start again, since the protocol uses Proof of Humanity as a Sybil protection mechanism.

Conversely, judges with a reputation score higher than ϵ_+ will receive a SBT which will certify their honesty and dedication: they will be considered *trusted*. All these SBTs are issued by the application itself.

Trusted judges have the possibility to contribute towards the governance of the platform. Thus, unlike classic dApps, everything is divided proportionally among all the users who have spent their time on the platform to make the service work. Indeed, we observe that in an application such as the one described in this paper there is no need to introduce any native ERC-20 token.

VI. Resilience to Attacks

In this section we outline the attack vectors that are prevented in our novel dispute resolution mechanism.

Voter coercion: voters cannot be coerced to vote in a certain way because their vote is encrypted and an adversary cannot deduce the content of the voter's ballot. Hence, the voter has no way of proving to a coercer how they voted, and thus a coercer cannot ensure that their victim complied. The only agent with access to a voter's content is the MACI coordinator, who shares a symmetric key with the voter to tally the votes. It is assumed the MACI coordinator is a trusted entity, and in future work alternatives to distribute tallying may be explored.

Vote selling: similarly, vote selling and buying is not a relevant concern, since the buyer cannot verify the content of their purchased vote. Voters may at most sell their private key to encrypt their vote, however, in doing so they forgo their

Phase	User A	User B	M Judges	Trusted coordinator	DRApp
Phase 1	1	1	M (one per judge)	-	-
Tally phase	-	-	-	1	-
Phase 2	1	1	-	-	-
Enforcement and SBT distribution	-	-	-	-	2

TABLE I
NUMBER OF TRANSACTIONS SENT BY THE ACTORS INVOLVED DURING THE PROCESS

identity, and therefore their ability to participate in any other election, because their key-pair containing their secret key is generated through the Semaphore group inclusion.

Voter information asymmetry: public votes create information asymmetry for voters participating in the election. The first voter to cast their vote knows no information about what will the election outcome be, conversely, the last voter to cast their vote can compute the election outcome themselves. This may be exploited by latter stage voters, and cause a herding effect in elections.

Double voting: a user cannot cast more than one vote as this is prevented through Semaphore.

Hostile take-overs: it is not possible for a voter to purchase large amounts of voting power in our protocol. Judges have one vote each, and disputing parties are assigned a fixed number of votes. Therefore, there is no way for an agent to amass sufficient voting power to win an election with only their vote. This is prevented in our system because qualified judges assign a finite amount of *voice credits* to each person in the dispute. Voters cannot purchase or otherwise acquire these *voice credits*.

Spam protection: the protocol presented is resistant to spam attacks, because to start a conflict a user must deposit a fee. The challenged party cannot ignore the dispute because otherwise they automatically lose it. There is no incentive to maliciously start a fake dispute as this would cause the user to lose their fee deposit. Similarly, there is no incentive to refuse entering a dispute because otherwise you're automatically the losing party. This means the winner's proposal is enacted.

Sybil protection: users cannot gain an unfair advantage by creating a large number of fake identities. This is prevented through the Proof of Humanity component in the protocol that ensures a user is a real human and that they have not already registered in the platform.

VII. IMPLEMENTATION AND COST DETAILS

We note that all the components necessary to implement this dispute resolution mechanism are already available and currently being used in the Ethereum ecosystem. A non-exhaustive list of projects that use them is the following:

- Proof of humanity is currently being used by Gitcoin [23], WalliD [37] and Fyre[13] amongst others.
- Semaphore has been used as a protocol in Unirep, Zupass and Bandada (for all of them, see [32]).
- Quadratic voting and funding are frequently used in Web3 applications. Example instance includes [30].
- MACI is currently part of clr.fund [7] and QFI [29].

We do not provide a specific implementation, as all individual components are already in use. The protocol we propose can be easily implemented on Ethereum. We proceed by outlining how each of these components interact with each other. Regarding Proof of Humanity and Semaphore, the zkPoH project [39] allows for the creation of Semaphore groups where the registered people must already be part of PoH. In terms of quadratic voting and MACI, MACI is already compatible with both quadratic voting and one-dollar-one-vote mechanisms. The interoperability of zkPoH and MACI can be addressed by simply registering the Semaphore public key pk into the MACI smart contract. This guarantees that only the registered users (i.e., judges) can interact with MACI. In other words, only users that are part of a specific Semaphore group earn the right to vote in MACI. Finally, the distribution of soulbound tokens is carried out through DRApp. This step does not require any custom solutions.

The proposal is generalisable to any other blockchain equipped with smart contracts; however, these tools, especially MACI, are currently only implemented in Ethereum. MACI can be replaced by any other on-chain voting mechanism, and does not need to be exclusive to Ethereum. Indeed, when a voting system emerges that formally satisfies Ballot Secrecy, that it can be used instead of MACI. Until then, the current system design is exclusive to Ethereum.

Table I recaps the amount of transactions sent by all the actors involved (users, judges, trusted coordinator and the DRApp itself) in a single dispute resolution process.

Observe that the maximum number of transactions sent by each user involved in the dispute is two, while every judge sends a single transaction. Hence, gas fees are limited.

The most expensive operation cost-wise is the computation of the MACI proofs. However, these proofs are calculated off-chain and the work is done by the MACI coordinator: even if this computation in general may require time (for example, the computation of MACI proofs in [22] required around 120 hours to perform the tally, but the number of voters was in the hundreds and there were 12 projects they could to vote for), we argue that in our context it should not require more than few hours, since the voters are the M judges and they vote for the two users involved in the dispute. In practice, exactly like Kleros, this number M will be a small odd number, for example $M = 5$ or $M = 7$. The time required to generate this proof for Groth16, the SNARK used by MACI, is $O(n \log_2(n))$ where n is the amount of operations required. Kleros is faster in returning the result of a dispute, but there is a waiting time to let parties appeal the result if they are unhappy with the outcome of the arbitrators. In that case,

	Kleros	Aragon	Our Contribution
Blockchain	Any with smart contracts	Ethereum - Aragon framework only	Ethereum
Judging	Requires a token stake	Requires a token stake	Requires a PoH account
Privacy	Partial, thanks to the use of commitments	Partial, thanks to the use of commitments	By default: MACI + Semaphore
Voting mechanism	One-person-one-vote + Schelling game	One-person-one-vote + Schelling game	One-person-one-vote + Quadratic voting
Resolution	Guaranteed, but judges have the last word	Guaranteed, but judges have the last word	Guaranteed, and conflicting parties have the last word
Incentives	Financial	Financial	Social compliance and financial

TABLE II
COMPARISON BETWEEN KLEROS, ARAGON AND OUR PROPOSAL.

Kleros dispute resolution process is performed again with a higher amount of arbitrators.

VIII. CONCLUSIONS AND FUTURE WORK

We have proposed a novel dispute resolution mechanism that provides a number of advantages over the state-of-the-art: Firstly, the judges resolving disputes are verified, real individuals and the voting system they use prevents collusion amongst them, as well as maintaining secret ballots to prevent them from being coerced. This is achieved using Semaphore and MACI, which guarantee the privacy of all users involved and, in addition, offer protection against Sybil and collusion attacks, respectively.
Then, participants in the conflict vote on which proposal they wish to enact, and the voting mechanism does not grant more power to the wealthier, but rather those deemed the most trustworthy (by the judges). Finally, there is no incentive to initiate spam conflicts, and good behaviour both by the participants and the judges is incentivised and rewarded by the system. Failure to reward the other side at the end of the dispute, as well as judges' misbehavior, is socially penalized through the use of Soulbound tokens that will be forever linked to their Ethereum wallet. In addition, judges who fall below a predetermined reputation score threshold are excluded from the system and will have no way to participate again. On the other hand, judges who exceed another predetermined threshold will have the opportunity to be part of the governance mechanism.

Table II compares Kleros and Aragon with our proposal. As we can see, our idea uses various protocols that guarantee privacy by default, unlike the existing applications. Furthermore, our system incentivises social compliance and honest behaviour, as well as creating economic incentives for participation. The protocol we propose can be used in the same use cases where Kleros is already being used. Examples include in disputes regarding token listings, escrow services, unfulfilled contractual obligations and grant funding.

a) ***Future Work****:* further research is needed to address the problem of "malicious but trusted" judges who try to exclude other trusted judges from the platform for their own interest. In addition, some components of our proposal are currently only available on Ethereum, so for future developments we aim to generalise the solution to applications beyond Ethereum. Developing an on-chain voting scheme that satisfies formal notions of ballot secrecy and verifiability is also necessary, as well as conducting a rigorous formal analysis of the security properties and incentive mechanisms of our protocol. Finally, implementing the mechanism would be useful to accurately analyse its performance compared to existing dispute resolution mechanisms.

ACKNOWLEDGMENTS

Andrea Gangemi is member of GNSAGA of INdAM and of CrypTO, the group of Cryptography and Number Theory of Politecnico di Torino. Andrea Gangemi is partially supported by the QUBIP project (https://www.qubip.eu), funded by the European Union under the Horizon Europe framework program [grant agreement no. 101119746] and SERICS (PE00000014 - https://serics.eu) under the MUR National Recovery and Resilience Plan funded by European Union - NextGenerationEU. Aida Manzano Kharman acknowledges and thanks IOTA Foundation for the funding of her Ph.D studies. The authors would like to thank Giacomo Corrias for his important comments and suggestions.

REFERENCES

[1] Ben Adida. Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348, 2008.
[2] Aragon. Aragon website, 2022. https://aragon.org/, Last accessed on 2024-08-16.
[3] Fadi Barbára, Andrea Gangemi, and Giulio Stefano Ravot. Overcoming the legal challenges of smart contracts through a "smart" commercial mediation. *Quaderni di conciliazione*, 24(2):271, 2020.
[4] Binance. Binance website, 2021. https://www.binance.com/en, Last accessed on 2024-08-16.
[5] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and Smart Contracts for the Internet of Things. *IEEE Access*, 4:2292–2303, 2016. doi: 10.1109/ACCESS.2016.2566339.
[6] Michael R Clarkson, Stephen Chong, and Andrew C Myers. Civitas: Toward a secure voting system. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 354–368. IEEE, 2008.
[7] Clr.fund. Clr.fund website, 2023. https://clr.fund/#/, Last accessed on 2024-08-16.

[8] Luis Cuende and Jirge Izquierdo. Aragon network a decentralized infrastructure for value exchange. *Available at SSRN 4105763*, 2017.

[9] Philip Daian, Tyler Kell, Ian Miers, and Ari Juels. On-Chain Vote Buying and the Rise of Dark DAOs, 2018. https://hackingdistributed.com/2018/07/02/on-chain-vote-buying/.

[10] Qinxu Ding, Weibiao Xu, Zhiguo Wang, and David Kuo Chuen Lee. Voting Schemes in DAO Governance. *Forthcoming in Annual Review of Fintech*, 2023.

[11] ERC1497. ERC 1497 Evidence standard, 2018. https://github.com/ethereum/EIPs/issues/1497, Last accessed on 2024-08-16.

[12] Yixuan Fan, Lei Zhang, Ruiyu Wang, and Muhammad Ali Imran. Insight into Voting in DAOs: Conceptual Analysis and A Proposal for Evaluation Framework. *IEEE Network*, 2023.

[13] Fyre. Fyre Documentation, 2023. https://fyre.id/, Last accessed on 2024-08-16.

[14] Jens Groth. On the size of pairing-based non-interactive arguments. In *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 35*, pages 305–326. Springer, 2016.

[15] IPFS. IPFS page, 2022. https://ipfs.tech/, Last accessed on 2024-08-16.

[16] Kleros. Kleros website, 2022. https://kleros.io/, Last accessed on 2024-08-16.

[17] Kleros dispute example 1. Curated list, 2023. https://resolve.kleros.io/cases/552, Last accessed on 2024-08-16.

[18] Kleros dispute example 2. Escrow, 2023. https://resolve.kleros.io/cases/561, Last accessed on 2024-08-16.

[19] Kleros dispute example 6. Gitcoin grant, 2023. https://resolve.kleros.io/cases/406, Last accessed on 2024-08-16.

[20] Steven P Lalley and E Glen Weyl. Quadratic voting: How mechanism design can radicalize democracy. In *AEA Papers and Proceedings*, volume 108, pages 33–37. American Economic Association 2014 Broadway, Suite 305, Nashville, TN 37203, 2018.

[21] Clément Lesaege, Federico Ast, and W George. Kleros. 2018.

[22] MACI proof computation time for clr.fund. Maci, 2022. https://mirror.xyz/crisgarner.eth/rvDF2Ppe7jd3W6U_RYkxxisuC1NZeApM9vgPoemunAM, Last accessed on 2024-08-16.

[23] Tara Merk, Sofia Cossar, and Jamilya Kamalova. Ethnographic research of proof of humanity dao (investigación etnográfica de proof of humanity dao). *Available at SSRN 4438414*, 2023.

[24] James Metzger. The current landscape of blockchain-based, crowdsourced arbitration. *Macquarie Law Journal*, 19(Nov 2019):81–101, 2019.

[25] OpenLaw. OpenLaw website, 2019. https://www.openlaw.io/, Last accessed on 2024-08-16.

[26] PoH. Proof of Humanity website, 2021. https://www.proofofhumanity.id/, Last accessed on 2024-08-16.

[27] Cristina Poncibò, Andrea Gangemi, and Giulio Stefano Ravot. Blockchain justice: Exploring decentralising dispute resolution across borders. *Journal of Law, Market & Innovation*, 3(1):14–32, 2024.

[28] PSE. Privacy and Scaling Exploration website, 2023. https://pse.dev/, Last accessed on 2024-08-16.

[29] QFI. QFI github page, 2023. https://github.com/quadratic-gardens/qfi#protocol-diagrams, Last accessed on 2024-08-16.

[30] Quadratic-funding. Gitcoin Grants – Quadratic Funding for the World, 2022. https://www.gitcoin.co/blog/gitcoin-grants-quadratic-funding-for-the-world, Last accessed on 2024-08-16.

[31] Semaphore. Semaphore website, 2022. https://semaphore.appliedzkp.org/, Last accessed on 2024-08-16.

[32] Semaphore projects. Semaphore projects, 2022. https://semaphore.pse.dev/, Last accessed on 2024-08-16.

[33] Ben Smyth. Ballot secrecy: Security definition, sufficient conditions, and analysis of Helios. *Journal of Computer Security*, 29(6):551–611, 2021.

[34] Nick Szabo. The Idea of Smart Contracts. https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_idea.html, 1997.

[35] Vitalik Buterin. MACI, 2019. https://ethresear.ch/t/minimal-anti-collusion-infrastructure/5413, Last accessed on 2024-08-16.

[36] Vitalik Buterin. Negative votes in quadratic funding, 2023. https://ethresear.ch/t/negative-votes-in-quadratic-funding/6855, Last accessed on 2024-08-16.

[37] Wallid. WalliD Documentation. https://docs.wallid.io/, Last accessed on 2024-08-16.

[38] E Glen Weyl, Puja Ohlhaver, and Vitalik Buterin. Decentralized Society: Finding Web3's Soul. *Available at SSRN 4105763*, 2022.

[39] zkPoH. Zero Knowledge Proof of Humanity, 2023. https://hackmd.io/@elmol/BkqjDy-k2, Last accessed on 2024-08-16.