Towards a Privacy-Preserving Dispute Resolution Protocol on Ethereum

***Abstract*—We present a new dispute resolution protocol that can be built on the Ethereum blockchain. Unlike existing applications, privacy is ensured by design through the use of zero-knowledge protocols, which provide resistance to Sybil attacks and corruption. A resolution to the dispute is guaranteed, whilst ensuring the users have the final say. Moreover, the proposed model rewards stakeholders through a social incentive mechanism based on Soulbound tokens. To our knowledge, this is one of the first proposals to implement governance through reputation as opposed to token-based voting.**

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

I. Introduction

In any Web3 application, conflicts and disputes of many kinds may arise. If the application is not providing the service as intended, users must have a means to initiate disputes. For instance: Binance [3] must ensure that all tradeable tokens on their site are related to valid Blockchain projects, and the Proof of Humanity protocol [27] must only add real humans to its registry. Indeed, if this were not the case, the consequences would be catastrophic.

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 [14].

However, all these dispute resolution services hare 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], as well as being private by design, and making collusion between judges 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, while Section VI explores possible attacks and how our protocol is able to resist them. Finally, Section VII outlines paths for possible future work.

II. Related Work

We present the survey of related work both in the context of existing conflict resolution mechanisms, as well as voting mechanisms used in Web3 dApps. Although our contribution is a dispute resolution mechanism, we require the use of a voting mechanism to enable its functioning. We explore existing voting mechanisms and introduce the one we select and why.

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 [14] 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, rather, it requires one for its own functioning. Aragon [1]

[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 [4][32].

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 [26].

We focus on the first category, since our proposal is designed to be a global mechanism that is compatible with any Ethereum DAO, but we also describe how it compares to other notable dispute resolution mechanisms such as Aragon and Jur [13] for context.

Kleros [14, 23] 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. It operates as follows: provided a dispute between two users arises, one party will initiate it, and then some time is given to the other party to join and contest that dispute. Refusal to join results in automatic loss of the dispute, resulting in automatic victory for the party who initiated the conflict. Both parties have to pay a fee in order to engage the service of a dispute resolution mechanism. Judges are randomly selected and then asked to vote amongst different options: the voting system implemented is the *plurality voting system*. The winning option coincides with the option that receives the supra-majority of votes. At the end of the process, only the winner of the dispute gets the fee back, while the loser fee will be used to reimburse the judges for their work. 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 [15–20].

Aragon [1, 5] is a DAO that enables the development and maintenance of decentralised organisations running on the Ethereum Virtual Machine. The Aragon Network is an internal organisation responsible for providing the infrastructure and a range of services to support users of the platform. Aragon token holders can access the Aragon DAO services, and one of these services is the Aragon Court, which tries to solve 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.

Finally, Jur [13] is a legal technology company that aims to create a legal ecosystem for the management of contractual relations. They are currently undergoing a transition phase, switching from the VeChainThor blockchain [33] to Polkadot [38]. They have yet to solve a dispute, but are one of the most promising projects in the space. They plan to use 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 [25]) and Aspera [2]. 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 Mechanisms

Next, we proceed by presenting and comparing the most commonly used voting mechanisms in Web3 dApps. In our proposal, we require the use of a voting mechanism, 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. This makes it a crucial component of the proposal. Research on voting mechanisms in the context of dApps is still young. We find that [7] and [10] propose criteria to evaluate the suitability of voting schemes, but we consider some of these criteria to be incomplete or inaccurate. [10] 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 proposed classification in [7] is also inadequate. Their *robustness* definition is also lacking any formalisation and their *fairness* definition 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*, we must consider formalised notions of privacy: a voter should have 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 [31]. Without *Ballot Secrecy*, voters may sell their votes, DAOs to buy votes (vote buying cartels) can arise [6] and smart contracts can be written to execute and automate transactions of votes for money. Indeed, the conditions for said exchange can be verified, because the coercer can check how you voted in the absence of *Ballot Secrecy*. We must also consider scenarios where voting power can be purchased: it must be prohibitively expensive for a malicious attacker to amass enough voting power such that they can take over the election. Finally, the voting scheme must ensure *verifiability*. Voters should be able to verify that the election outcome does indeed represent the voters' votes. The work in [30] identifies that individual verifiability[3] and universal verifiability[4] plus the cast-as-intended[5] property does not yield verifiable voting systems, and that further work must

[3]A voter can check whether their ballot has been included.

[4]Anyone can check that the tally of votes reflects the votes expressed in collected ballots.

[5]The cast ballot contains the vote that the voter wishes to express.

be done to build a suitable security notion. This is beyond the scope of our research, thus we assume that individual and universal verifiability suffice.

We consider the following criteria:

Fairness: the voting mechanism does not give some voters more power than others unintendedly.

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

Security: 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 [31].

a) ***Permissioned Majority Voting:*** The most simple and commonly used voting mechanism is Permissioned Majority Voting. In it, only token holders are allowed to vote. If a proposal receives more than half the necessary votes, it is approved. It is simple to compute the outcome and easy for participants to understand. However, it suffers from an important drawback: since voting power is often acquired through tokens, a single malicious and wealthy agent may easily take over the election. It suffices for them to amass enough tokens such that with their own vote alone, they can approve any (including their own) proposals. Majority Voting, also known as Plurality Voting, is known to suffer of many disadvantages. Namely: groups that use this system converge to a two-party system [11], it is susceptible to tactical voting [8], and it may affect voter turnout [21], since a winner-takes-all system does not encourage voters to vote against the most preferred option. This latter drawback would be especially exacerbated in the context of our work, given that to vote, voters must spend their tokens. In conclusion, the system is fast, but does not treat all voters equally (despite what [7] claims).

b) ***Conviction Voting:*** In conviction voting, participants cast their vote for or against a proposal, and the longer the vote remains unchanged, the more conviction (voting power) that vote is given [7]. Proposals are approved if they receive enough levels of conviction. This mechanism is fairer than Majority Voting, since it takes into account the wider beliefs of the community and prevents rapid acquisition of voting power. However, it is time-consuming to run.

c) ***Token-based Quorum Voting:*** In Token-based Quorum Voting a minimum number of members must take part in the voting, and if this condition is satisfied, the proposal with the most votes gets enacted. Without the quorum, the proposal will fail [10]. A number of issues arise with this approach. Participation levels can negatively impact the approval rate of proposals that get approved. In situations where the proposal is an urgent bug-fix, this can have catastrophic consequences, as the vulnerability remains exploitable until a patch is approved. Furthermore, certain agents may purposefully sabotage proposals by abstaining. Lowering the quorum level can circumvent the aforementioned issues, but this comes at the expense of security and decentralisation.

d) ***Quadratic Voting:*** Similar to Conviction Voting, this mechanism allows voters to express the amount of conviction in their vote. Instead of amassing voting power by allowing for time to elapse, voters may acquire more than one vote and cast it. However, the cost of acquiring more votes is not linear, but rather quadratic. The more voting power you obtain, the harder it is to acquire more of it [22]. This 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. This is not a relevant concern in the context of our proposal, because voters have a limited amount of wealth they can spend on votes, 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, and fairer than Majority Voting and Token-based Quorum Voting, since it makes monopolisation of voting power prohibitively expensive. Given this compromise, we select this voting mechanism for our proposal, and further justify this decision in Section III.

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 users to prove possession of some information, without revealing said information.

Then, we outline the functioning of *Proof of Humanity* and how it can be integrated with Semaphore. Subsequently, we introduce *quadratic voting*, and justify its use over other existing voting mechanisms. 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:*** *Semaphore* [29] 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. More in detail, Semaphore provides three functionalities:

- *Creation of private identities.* A user that joins a Semaphore group receives a secret/public key pair $(\mathsf{sk}, \mathsf{pk})$. More precisely, the secret key is a tuple of three values $\mathsf{sk} = (\mathsf{IdTrapdoor}, \mathsf{IdNullifier}, \mathsf{IdSecret})$, where $\mathsf{IdTrapdoor}$ and $\mathsf{IdNullifier}$ are generated randomly, while $\mathsf{IdSecret} = H(\mathsf{IdTrapdoor} \,||\, \mathsf{IdNullifier})$ and H is a hash function. The nullifier is needed to avoid users signaling more than once. The public key is instead the hash of the quantity $\mathsf{IdSecret}$: $\mathsf{pk} = H(\mathsf{IdSecret})$. 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.

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*. Signals are signed messages which are 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 just signal once. For this reason, each signal contains also two additional values: a public one, $\mathsf{ExtNullifier}$, which is usually the ID of the Semaphore group, and then the digest $\mathsf{Nullifier} = H(\mathsf{IdNullifier} \,||\, \mathsf{ExtNullifier})$. Since $\mathsf{IdNullifier}$ is part of the private key of a user, if two different signals have the same value $\mathsf{Nullifier}$ 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 [29].

b) ***MACI:*** MACI [24] stands for *Minimal Anti-Collusion Infrastructure* and it is a protocol that allows users to vote on-chain with a greatly increased collusion resistance. It was proposed in [34]. All transactions on a blockchain are public, so a voter can easily show to a briber which option they voted for. MACI counters this issue by using zk-SNARKs to hide how each user voted, while still providing the result of the tally of the votes.

The MACI protocol has two different actors: *users*, the entities that send a vote, and a single *trusted coordinator*, which counts the votes and releases the final result. The coordinator uses zk-SNARKs to prove that the tally has been correctly computed, using only valid votes, without revealing the vote of each user.

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 can be spent to vote on some proposal. They can vote with any address, but the transaction that expresses their point of view must contain an information about 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 the transaction data a different public key $\widetilde{\mathsf{pk}_I}$, while 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, a transaction, to be considered valid, 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 whether or not 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 keep track of all the valid votes. Then, it does 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 messages, 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 their individual contribution leads to the final result.

Both proofs are verified by another smart contract SC_4, specifically built to read MACI proofs.

However, it must be kept in mind that MACI has one major flaw, namely, it relies all its security on a central authority. If the latter were compromised, the whole protocol would fail in its intent. One possible solution might be to use an approach derived from *multi-party computation*, and thus have N authorities instead of just one. In this case, the system is secure if some subset of the N parties is honest, however the currently implemented MACI protocol does not offer this possibility.

B. Proof of Humanity

Proof of Humanity [27] (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. Some 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* [22] 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 may be interesting to also allow *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 [35].
As the next proposition shows, quadratic voting is a better solution compared to 1D1V, when voters have a different number of voice credits to assign.

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
$$(\sqrt{V_A} - \sqrt{y_1}) < \sqrt{y_2}, \qquad y_1 + y_2 = V_B.$$
With some manipulations, 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 [37]. 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, but, given an application, we can achieve the "right" degree of privacy with a mix of cryptographic protocols like zero-knowledge proofs.
Soulbound tokens can be given 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 its users to have the final say on the resolution of the conflict, i.e. they should not be forced to accept the judges' decisions;
2) allow potential judges to participate without requiring them to stake tokens to vote on a dispute resolution, as not everyone can afford it;
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 not to those users who possess a high number of tokens, but rather 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. Leveraging the privacy properties of MACI, 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 we need the MACI protocol to 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 now that 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 [12], or *Web3 storage* [36]. Another alternative is the mechanism introduced by Kleros and explained in the ERC-1497 proposal [9].
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 some level of decentralisation.
Every judge will vote using its 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.
At the end of the process, the proposal that received the highest score is enforced.

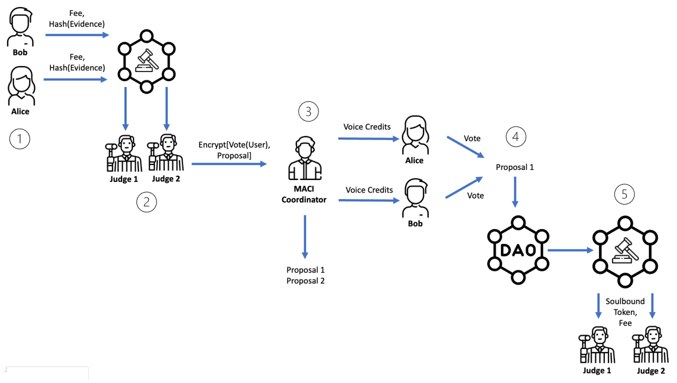

Fig. 1. Dispute Resolution Mechanism. Credit for icons given in:[6]

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 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.

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

3. The MACI coordinator tallies the votes, which are assigned to each party as voice credits, and outputs the list of proposals.
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.

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 [37], the ultimate goal is to entrust the DAO governance to judges who have spent their time on the proper functioning of the platform, instead of giving it 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 at the passage of this time t 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.

B. Incentives for judges

Unlike Kleros, judges do not need a token stake to participate to a dispute. However, we still need a way to incentivise good behavior and especially to penalize bad behavior, otherwise a judge can simply register and then vote effortlessly 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 without paying attention, 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 even create a new account and start again, since the protocol uses Proof of Humanity as a proof-of-personhood 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 identity, and therefore their ability to participate in any other election, because their key-pair containing their secret key is

	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 I
COMPARISON BETWEEN KLEROS, ARAGON AND OUR PROPOSAL.

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 can be exploited to mount *'pump and dumps'*. A proposal may be put forth that can stand to increase the value of a DAO token. If votes are public, participants can observe that that proposal is set to be enacted if it is amassing votes. Participants may rush to acquire more of said token, but a voter that has a large amount of *voice credits* can withhold their vote until just before the election ends. They sell their tokens just before the end of the election, cast their vote against the proposal and the token value plummets. They have made a profit unlike all other participants. Our protocol resolves this by keeping votes secret, no external observer or voter can deduce the content of the votes as the election is taking place.
Double Voting: A user cannot cast more than one vote as this is prevented through Semaphore.
Forcible take-overs: It is not possible for a voter to purchase large amounts of voting power in our protocol. 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, and no incentive to avoid entering it because otherwise you're automatically the losing party, and the winning party's proposed dispute resolution 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. 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 two zero-knowledge protocols, 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 I 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.

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. One possible solution could be the initiation of a dispute by the rest of the network of trusted judges towards the malicious judge. In addition, given the greater complexity of the protocol compared to the one proposed by Kleros, it is necessary to estimate the total costs of the gas used in order to understand, from a monetary point of view, when it makes sense to use the new model rather than theirs.
Finally, 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.

References

[1] Aragon. Aragon website, 2022. https://aragon.org/, Last accessed on 2023-11-17.

[2] 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.

[3] Binance. Binance website, 2021. https://www.binance.com/en, Last accessed on 2023-11-17.

[4] 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.

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

[6] 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/.

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

[8] Bernard Dolez, Annie Laurent, and André Blais. Strategic voting in the second round of a two-round system: The 2014 French municipal elections. *French politics*, 15:27–42, 2017.

[9] ERC1497. ERC 1497 Evidence standard, 2018. https://github.com/ethereum/EIPs/issues/1497, Last accessed on 2023-11-17.

[10] 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.

[11] Bernard Grofman, André Blais, and Shaun Bowler. *Duverger’s Law of plurality voting: The logic of party competition in Canada, India, the United Kingdom and the United States*, volume 13. Springer, 2009.

[12] IPFS. IPFS page, 2022. https://ipfs.tech/, Last accessed on 2023-11-17.

[13] Jur. Jur website, 2022. https://jur.io/, Last accessed on 2023-11-17.

[14] Kleros. Kleros website, 2022. https://kleros.io/, Last accessed on 2023-11-17.

[15] Kleros dispute example 1. Curated list, 2023. https://resolve.kleros.io/cases/552, Last accessed on 2023-11-17.

[16] Kleros dispute example 2. Escrow, 2023. https://resolve.kleros.io/cases/561, Last accessed on 2023-11-17.

[17] Kleros dispute example 3. Insurance, 2023. https://resolve.kleros.io/cases/548, Last accessed on 2023-11-17.

[18] Kleros dispute example 4. Token, 2023. https://resolve.kleros.io/cases/547, Last accessed on 2023-11-17.

[19] Kleros dispute example 5. Social networks, 2023. https://resolve.kleros.io/cases/145, Last accessed on 2023-11-17.

[20] Kleros dispute example 6. Gitcoin grant, 2023. https://resolve.kleros.io/cases/406, Last accessed on 2023-11-17.

[21] Agnieszka Kwiatkowska and Mikołaj Cześnik. Electoral System, Political Knowledge and Voter Turnout—Complex Liaisons. *Polish Sociological Review*, (212):425–444, 2020.

[22] 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.

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

[24] MACI. MACI GitHub page, 2022. https://github.com/privacy-scaling-explorations/maci, Last accessed on 2023-11-17.

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

[26] OpenLaw. OpenLaw website, 2019. https://www.openlaw.io/, Last accessed on 2023-11-17.

[27] PoH. Proof of Humanity website, 2021. https://www.proofofhumanity.id/, Last accessed on 2023-11-17.

[28] PSE. Privacy and Scaling Exploration website, 2023. https://pse.dev/, Last accessed on 2023-11-17.

[29] Semaphore. Semaphore website, 2022. https://semaphore.appliedzkp.org/, Last accessed on 2023-11-17.

[30] Ben Smyth. Mind the Gap: Individual- and universal-verifiability plus cast-as-intended don’t yield verifiable voting systems. Technical Report 2020/1054, Cryptology ePrint Archive, 2020.

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

[32] 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.

[33] Vechain. VechainThor website, 2023. https://www.vechain.org/, Last accessed on 2023-11-17.

[34] Vitalik Buterin. MACI, 2019. https://ethresear.ch/t/minimal-anti-collusion-infrastructure/5413, Last accessed on 2023-11-17.

[35] Vitalik Buterin. Negative votes in quadratic funding, 2023. https://ethresear.ch/t/negative-votes-in-quadratic-funding/6855, Last accessed on 2023-11-17.

[36] Web3 storage. Web3 storage page, 2022. https://web3.storage/, Last accessed on 2023-11-17.

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

[38] Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework. *White Paper*, 21:2327–4662, 2016.

[39] zkPoH. Zero Knowledge Proof of Humanity, 2023. https://hackmd.io/@elmol/BkqjDy-k2, Last accessed on 2023-11-17.