Scaling Idea for Kleros Court

Summary: This is a first idea for how (low fee subcourts of) the Kleros court might scale by minimizing the amount of required on-chain information. This proposal still keeps some information in on-chain transactions that ideally could be eliminated; however it largely avoids issues related to data availability. This idea uses (a higher value on-chain subcourt of) the Kleros court itself to resolve potential disagreements over the status of the side-chain subcourt.

1 On-chain Elements

1.1 Types of On-chain Transactions

One should be able to interact with the side-chain court contract by raising disputes coming from arbitrable contracts and staking PNK to signal availability to be drawn as a juror as is the case for current courts. Additionally, we look in greater detail at four other types of on-chain transactions that interact with the court contract.

- Aggregation (State Changes) - these transactions periodically report important information from the side-chain court on-chain. This reporting is similar to the work of a Plasma operator, though note that in this setup anyone can submit aggregation transactions. The content of these transactions is:
  - The results of some number of (finalized) appeal rounds of cases - i.e. something of the form: case number, round number, result
  - Some number of destake balances of the form: Ethereum address, ETH that can be withdrawn, PNK that can be withdrawn
  - Blockhash of a recent block (this is to prevent certain replay attacks, see below).
- Aggregation (Data Availability) - Similar to Aggregation (State Changes) these transactions report information from the side-chain on the main-chain; however they do not change the state in doing so. Rather, these transactions are used for to demonstrate the availability of data that (after some delay) Kleros jurors can assume is known to issuers of other transactions.

  - New votes on cases. No logic will be applied to this/it doesn't necessarily need to be in a format understandable by the EVM. As such it can conceivably be placed in calldata à la rollup. Could be of form: disputeID, appeal, voterID, vote. (Unlike the results of the cases and the destake balances, this does not a priori need to be on-chain to have a viable version of Kleros that arbitrable Dapps, potentially on other shards, would be able to interact with. Possible alternative ideas to not have these transactions, at the expense of requiring jurors to judge data availability as part of their decision making process and having greater data availability issues.)

- ChallengeData - these transactions allow one to challenge a malicious Aggregation (Data Availability). The content of these transactions is:
  - Aggregation (Data Availability) to challenge
- Destake - these transactions allow jurors to signal their intention to destake from the side-chain court. They will only be able to actually withdraw their PNK from the court contract after their balance is confirmed in an aggregation that goes through a challenge period without being challenged. The content of these transactions is:
  - Address to destake from
  - (Maximum) amounts of ETH and PNK that want to destake

Note that we will have several global parameters (which are generally very TCRish). These include:

- Challenge period for aggregations
- Deposits required to submit/challenge aggregation
- Potentially some percentage of fees for the votes that are included in the aggregation, the amounts that are being destaked, to incentivize aggregators.

1.2 On-chain logic

In this section we describe the logic around these transactions that should be enforced on-chain. First, we describe the conditions under which the different types of transactions are reverted:

Aggregation (State Changes):

- Revert if formatted incorrectly or inadequate bond provided
- Revert if the included recent blockhash does not correspond to the current chain.

Aggregation (Data Availability):

- Revert if formatted incorrectly or inadequate bond provided

ChallengeData:

- Revert if formatted incorrectly or inadequate bond provided
- Revert if specific Aggregation (Data Availability) is already challenged

Destake:

- Revert if formatted incorrectly.

Then,

- If there is only one Aggregation (State Changes) submitted during its challenge period, then the reported results can be used by other Dapps, users can withdraw the amounts approved as destake-able. The aggregator receives her deposit back + any fees. (Parts of this might require some additional finalization transaction.)
- If there is more than one Aggregation (State Changes) in a given period, this raises a Kleros court case. At stake are the deposits of the submitters of the two aggregations. The winning aggregation is implemented, namely its reported results can be used by other Dapps and users can withdraw the amounts approved as destake-able[1]. In case no aggregation is winning, i.e. jurors rule for "None of the above", then no results or destakes are implemented for this period (however, these can be resubmitted in the current period, see below).
- If an Aggregation (Data Availability) goes unchallenged during its challenge period, the aggregator receives back her deposit.
- If an Aggregation (Data Availability) is challenged, this raises a Kleros court case. At stake in this case are the aggregator and challenger deposits.

2 Side-chain elements

2.1 Scaling Court Policy, i.e. "Side-chain Logic"

Jurors in an on-chain "Scaling Court" should do the following to simulate the state of the "side-chain" (i.e. votes on cases, balances of participants in ETH and PNK) based on the information that is available to them on-chain:

- Track how much PNK each participant in the side chain has staked via the on-chain stake and destake transactions, adjusting for won or lost PNK from cases on the side-chain that are finalized. For the purposes of determining whether a juror is coherent or not in simulating the calculation

[1] There might be a period after submission where an aggregator can cancel her aggregation in case some other similar aggregation has been submitted at the same time, similar to the current functioning of the Kleros governor: https://governor.kleros.io/

of her current stake, one can consider her vote to be the earliest report of her vote in an Aggregation (Data Availability) transaction that is not challenged during its challenge period[2] and is within the case's voting period. (Note here that when a juror is casting her vote in the Scaling court, there is a specific history of previous blocks leading to the block that selects her. Hence the inclusion or exclusion of stake and destake transactions is well-defined according to this chain history. Similarly, the inclusion of past, unchallenged aggregations that lead to finalized results of cases is also well-defined. Note, however, if Ethereum had a long fork, the "honest" vote in a case might depend on which side of the fork one was on. Then ideally Kleros would be resistant to "replay attacks"; i.e. that a vote cast intended for one side of the fork would not be valid on the other.)

- Observe on-chain the blockhash that is generated following the creation of the dispute by the arbitrable contract that is used to provide randomness to the juror selection process. Then jurors in the Scaling court can simulate the deterministic process that is used from that point to determine which Ethereum addresses should be jurors for a given case/appeal round.

- Determine the current vote count in each dispute by taking a juror's vote to be the earliest report of her vote in an Aggregation (Data Availability) transaction that is not challenged during its challenge period and is within the case's voting period.

- Further, track fees paid to aggregators (if the fee parameters are set to non-zero values). For example, an aggregator might receive a percentage of the arbitration fees for each vote that she includes in an aggregation, such that this is the earliest aggregation that reports a value for that vote, the aggregation is within the case's voting period, and the aggregation transaction is not challenged (then, if two aggregators include the same vote in their transactions, only the first will receive the fee for that vote, while the second aggregator may nonetheless receive fees for some other votes she includes if they had not been previously aggregated). Aggregators could be similarly rewarded for reporting results and destake-able balances in Aggregations (State Changes). These values accumulate to a balance that can be destaked and withdrawn from the side-chain similar to that of jurors.

[2]Particularly, we want to avoid having to have Aggregation (State Changes) transactions wait for the results of Kleros cases over challenged Aggregation (Data Availability) transactions. If Aggregation (Data Availability) transaction is challenged maliciously, its aggregator, confident in their ability to win an eventual Kleros court case and get a compensation from the challenger's deposit, can resubmit the same information in a new transaction. While this leads to bank attacks of hostile parties challenging all Aggregations (Data Availability) transactions with votes against them, accepting that attack model was already largely inevitable, see the discussion of bank attacks in Section 3. On the other hand, this is in contrast to the Aggregation (State Changes) transactions where there is automatically a dispute if there is more than one in a given period, due to the greater security risks of approving malicious destakes.

Then, in case of a ChallengeData, an Aggregation (Data Availability) should be considered to be invalid if:

- The aggregator is incapable of providing a signature corresponding to a vote that she included[3].

In case of a Kleros court case arising from multiple Aggregation (State Changes) transactions being submitted in the same submission period, jurors should consider an Aggregation (State Changes) transaction to be invalid if any of the following:

- A result is reported before its reporting period begins.
- A result does not correspond to the vote count, as tracked in the way described above.
- The destake balance does not correspond to an amount to which a participant is entitled (whether for jurors or for aggregators) as described above.
- A destake balance is provided for which no destake transaction was issued, or a balance is provided that exceeds the amount requested to be destaked.

Then jurors should vote in favor of the first valid Aggregation (State Changes) transaction received. In case none of the Aggregation (State Changes) transactions are valid, jurors should vote in favor of a "None of the above" option and the results and destakes of this period are eligible to be included in the current period/in a future period.

Note that here the voting rule and the redistribution rules for these "Side-chain courts" are all enforced at the level of the Scaling Court policy that determines, rather than directly in smart contracts. Also remark that the verification process done by the jurors to determine if aggregations are valid could in theory all be done via some kind of interactive game on-chain; however, this process would still require delays to give aggregators time to provide requested

[3]Normally, we envision signed transactions being widely available in some off-chain infrastructure, see Section 2.2. However, if a vote is aggregated for a transaction which does not *seem* to be available, one has an instance of the data availability problem. Jurors in the scaling court could potentially rule on data availability as a condition for aggregations being valid; however this risks making their task less clear cut. In contrast, all other data that jurors need to simulate the current state of the side-chain is available through the various on-chain transactions. However, compare this structure with the types of exit games that are often considered in Plasma proposals, where a user often has to be lively enough to challenge malicious actions by the Plasma operator within a challenge period. Here, the liveliness requirements on jurors are that they challenge malicious aggregators that falsely report their vote within the voting period; indeed while an honest juror will be certain of the history of how she has signed votes and will know that she can win a challenge against an aggregation with a different vote history, no other user can know that the voter is not malicious and hasn't signed conflicting votes. However, Kleros jurors must already be online within the voting period in order to vote, so these liveliness requirements are not necessarily as problematic in our setting. See Section 3 for further discussion on this point.

signatures. Hence, such an interactive game wouldn't necessarily lead to a better UX than using a Kleros case and would likely require greater development resources. So, as we already rely on the security of the Kleros general court, this approach seems acceptable, at least for Kleros.

2.2 Side-chain Infrastructure

As described above, aggregators, challengers, and jurors should already have enough information to perform their roles. However, in order to enable their tasks, we probably want to have the following infrastructure in place:

- Signed votes on an IPFS or similar so that available and easy for aggregators to assemble.
- Some kind of tools in a UI that tracks everything to help aggregators, challengers, and jurors. Basically being a juror in the Scaling court requires being a sort of human node/running a node for the side-chain, so a client needs to be available to allow them to reasonably do that.
- Email notifications for situations where votes are included in Aggregation (Data Availability) transactions that are not available on the IPFS (namely which could have been maliciously reported by the aggregator). Also jurors should be notified if one is close to the deadline, and their vote hasn't been aggregated yet (giving them the time to aggregate their own votes if necessary, essentially equivalent to casting vote on-chain).

3 Limitations

In this section we conclude with some limitations and issues of this proposal as compared to the current fully on-chain courts.

- Depends on jurors being, at least *usually*, willing to pay the deposit to challenge a malicious aggregation that steals their vote.
  - Note that Plasma-type setups often depend on chain participants watching the chain to make sure that their funds are not being stolen. Here you can only falsely report someone's vote if they were chosen to vote in the case in the first place. Moreover, jurors already have to take an action with in any given vote period, so they will presumably often notice if they try to vote and someone has already created an aggregation with a false transaction. So jurors do not need to be online under this setup (much) more than they already do. Honest jurors who see such a malicious aggregation know that they can win any eventual challenge; however there is still the added friction of having to pay a deposit to start a challenge.
  - A few stolen votes here and there is not as potentially catastrophic as a malicious destake balance being reported, which could be used

to drain the side-court of its funds. Correspondingly, this proposal is designed in such a way that *all* participants have enough information to flag malicious destake balances. Nevertheless, one wants stolen votes to be flagged often enough in malicious aggregations so as to provide a deterrent for the aggregators.

  - This issue is particularly problematic if we expect a high enough rate of jurors not voting that attackers have decent odds of stealing the vote without being noticed.

- Depends on the integrity of the Scaling court to work properly
  - Particularly, jurors have to be capable of essentially being human nodes (or alternatively depend on a set of tools to help them in this) to follow a set of rules for how the side-chain operates
  - Also, note that a 51% attack on the general court allows the attacker to actually steal PNK from the side-chain courts, as she can cases regarding false Aggregation (State Changes) transactions to approve withdrawals to herself.

- While chances of being drawn in the side-chain court can adjust automatically to account for PNK won or lost, it is not clear how to have jurors' stake in the higher appeal courts/the general court adjust for the PNK they won or lost on the side-chain court, particularly without updating the current court contract.

- Worse UX
  - Won't have on-chain confirmation that vote included as quickly.
  - If vote of a given juror is being censored, she can submit an aggregation with her own vote; however, this requires putting forward the aggregator deposit.
  - Even, if a juror believes that her vote has made it on-chain because it has been included in some aggregation, that aggregation may later be challenged without the juror being aware. While the vote could be picked up by another aggregator, probably important to have good notifications to handle such situations.

- New types of bank attacks
  - Attackers can challenge all Aggregation (Data Availability) transactions in an attempt to prevent any votes for other side from going through.
  - In this case, there would be a cost for the attacker in terms of lost deposits, though it is probably inappropriate for large value cases to be on such side-chains.

  - Possible to delay results being reported and destakes by submitting malicious Aggregation (State Changes) transactions that require time to resolve.

- Not obvious how to make this proposal compatible with commit and reveal.
  - Could have same optimistic setup as above with regard to aggregators needing to be capable of producing a revealed vote that corresponds to a commitment, but not clear to where jurors would commit their votes if not on-chain.
  - As this proposal is specifically for cheaper courts, probably not so bad as can appeal these decisions to on-chain courts with commit and reveal if necessary.

- Potential for aggregators to be front-run by entities that are not spending the effort to actually follow the consensus rules of the side-chain but manage to obtain the aggregator fees.

- Long Ethereum forks can lead to replay attacks on jurors
  - Can have cases where "honest" vote in a dispute depends on which fork one is on (because different jurors should be drawn in the side-chain courts or aggregations do or do not make it in in the two forks). Then a juror might want to vote one way on one fork and differently on the other.
  - Similar issues for Aggregation (State Changes) transactions. Can be mitigated by including reference to recent block in transaction so that transaction reverts if on the wrong fork.
  - For jurors, less clear if can avoid these issues without updating the current court contract.