Changes to Next Version of Court

Major changes, some choices on these points to be finalized

- Court deployed on optimistic rollup, using relayer model to return results to L1/other rollups according to: https://github.com/kleros/research-docs/blob/master/research-notes/scalingcommunicationcourtstogether2.pdf
    - Realitio type disputes/disputes that don't have a natural plaintiff/challenger structure get a different set of relayers that entail somewhat more friction, final decision on what model to use for these still to be made from among choices presented in above doc; however most natural choice is probably the "bounty" approach presented on p.15
- New voting/incentive system - proposed choice is detailed on p.17 of this document https://github.com/kleros/research-docs/blob/master/research-notes/mcupdatednotes.pdf if seems problematic for any reason, e.g. dev feedback suggests difficulties in implementation, etc, alternative voting/incentive systems possible based on tradeoffs detailed in above document (keep in mind that will be on Optimistic Rollup, so gas costs calculated in terms of calldata costs and expected costs to use that network, so one still needs to pay some attention to what is acceptable in terms of gas)
- Forking mechanism, more or less as detailed[1] in section 4.9 of yellowpaper: https://kleros.io/static/yellowpaper_en-28d8e155664f3f21578958a482f33bd1.pdf
    - This comment: "We envisage writing arbitrable contracts in such a way that, with unanimous consent of the concerned parties, an arbitrator for an existing contract can be replaced with a fork of Kleros. Hence, this limits the ability of a successful 51% attacker to hold Kleros users hostage except in the relatively rare situation where the attacker has a direct interest. User interfaces can alert concerned parties to the fact that there has been a fork and over what case this fork was made." - only really makes sense for a limited number of applications, such as the escrow and Linguo. Decide whether worthwhile to implement in them.
- Changes to crowdfunding:
    - Put crowdfunding onto arbitrator side so that final, enforced decision (in situations where one side is funded and other wasn't) can be taken into account when jurors are rewarded/penalized
    - In cases where the last juror vote and the final outcome disagree, take one of following approaches (depending on code complexity, etc):
        - If the last juror votes and the final outcome disagree, treat these two alternatives as if tied. In particular, jurors who rank either of these alternatives first do not lose any of their PNK deposits. Lost deposits from other jurors and fees distributed accordingly.

[1] One point that needs to be altered/where a small error should be corrected is where Lj is defined as "Take the list Lj of voters USRi for whom aj is listed as an acceptable choice for some rij." on p.30. One should also require that USRi does not list a_main as an acceptable choice.

        - Alternative, (potentially simpler approach), in this case there should be no PNK penalties/redistribution, fees are still rewarded according to the final vote winner
    - Update to crowdfunding model to be compatible with multiple choices. Several choices possible that are described on p. 23-24 of the yellowpaper https://kleros.io/static/yellowpaper_en-28d8e155664f3f21578958a482f33bd1.pdf
        - Best choice from a game theory perspective seems to be the third one that begins “Suppose that option b won the previous appeal round. Then an appeal is called if ....” with gamma(k)=(k+1)/2.
        - If that implementation is too complicated, the other approaches would be acceptable.
    - If a given round’s result doesn’t have winner/loser (e.g. in case of a tie, etc) block crowdfunding contributions if at least one side isn’t fully funded at end of LoserFundingPeriod. If at least one alternative is funded by this point, allow the remaining time for crowdfunding other alternative(s). This avoids situations where one can fund at the last second to snipe a result.
    - Provide an incentive to funders who fund alternatives that win by default in the last round due to other alternatives not being funded. This compensates these funders for transaction costs they might take, as well as costs due to capital lockup, etc.
        - Let z be a new parameter that will represent an ETH value, and let delta be a new parameter that will represent a percentage.
        - When triggering a dispute, the fee required by the arbitrator contract is now $base\ arbitration\ fee\ paid\ to\ jurors + z$. (Arbitrable apps that require fees of arbitration fees times a stake multiplier should now take this adjusted amount in place of arbitration fees.)
        - Then, suppose that the alternative A is funded and then wins by default due to no other alternatives being funded; however, B !=A was the alternative that won the previous juror vote. We perform:
            - Summing over the funders in the previous crowdfunding round or in all previous crowdfunding rounds[2], the amount that a crowdfunder who 1) funded A in the previous round(s) and 2) did not fund B in the previous round(s)[3], reduce the payoff they would otherwise receive for correctly funding A by a factor of delta.
            - Pay the crowdfunder(s) of A in the last round $z + delta * (amounts\ described\ in\ previous\ step)$
        - If the final result is given by a juror vote rather than an undisputed crowdfund, the value of z can be added to the amounts to be

[2] Whichever is easier to implement, parameters can be adjusted accordingly.
[3] Note in one of the multiple choice crowdfunding models, a funder can fund on behalf of a set of alternatives. Hence, we draw the incentive specifically from the crowdfunders who benefited from the undisputed last-round funding of A by going from losing a stake by funding a non-winning alternative to being rewarded for having funded the winning alternative.

periodically airdropped to jurors staked in the court, see the discussion on the case when there are no coherent jurors below.

- Parameters that autoadjust - allow the parameters alpha (the percentage of a minstake that can be lost per vote) and feePerJuror to vary according to market conditions, using price oracles and information on gas costs. (NB: be careful in implementation of on-chain oracles to avoid attacks where someone takes a flashloan to temporarily mess with parameters, for example right before their case is drawn - if use an out-of-the box solution for price oracles verify that they are handling this well. If we do something ourselves based on Uniswap data, etc make sure to use best practices/long running average of recent data so resist flash loan manipulation.) Specifically, alpha*minstake should be a constant value in fiat terms and feePerJuror should be a constant value in fiat terms + average required gas. Then these target constant values should be adjustable by governance.
- Antiprerevelation: (Recall that dealing with antiprerevelation is a problem with two main dimensions. 1) Facilitating revealing of committed votes in a way such that using commit+reveal isn't too problematic for UX and 2) Incentivizing jurors to not just reveal their vote anyway.)
    - Add a new parameter that adds a value to arbitration fees, providing a small bounty for each vote that is submitted during the reveal period[4].
    - Creation of an "antiprerevelation court". In courts where commit+reveal is activated, in addition to the alpha*minstake that jurors can lose on a given incoherent vote, they have an additional locked amount of beta*minstake, where beta is a new parameter. A challenger can flag a vote as having been pre-revealed.[5]

---

[4] Frontrunning will obviously be an issue here for people actually collecting these bounties. The idea would be for the bounty to be at least sufficient to cover the transaction costs of third parties submitting revealed transactions. At launch, these values may all be set to zero absent a mechanism that would allow for jurors' votes to reveal during the reveal period without further action from them; however during the lifetime of this court contract, we expect that there will be such a mechanism such as add-on that can be installed on user devices to automatically reveal salts at a certain time, or ideally a system based on Pederson threshold encryption.

[5] This mechanism, in addition to leaning into a Kleros-centric approach, provides the flexibility that comes with having human beings judge pre-revelation, which can take very diverse forms, from simply publishing a salt, to issuing zk-proofs, etc, and hence can be challenging to control without some aspect of human oversight. Further, the primary document defining pre-revelation can be adapted based on need and tradeoffs around limiting pre-revealed votes versus allowing for deliberation on a case in Telegram groups that might include jurors on a case. If necessary, comments from a well-known Telegram account could be considered sufficient to be judged for pre-revelation. Note, however, that this mechanism is still imperfect. For example, an attacker could issue a proof using a traceable ring signature scheme that she is a juror who voted for Alice without revealing which juror she is, and hence not exposing herself to risks of losing a deposit due to a pre-revelation court case. Note that antiprerevelation games would also be vulnerable to such an attack. Indeed, the only known schemes that would resist such an attack are those based on anti-collusion resistant architecture, which in particular uses the idea that voters can resubmit votes over the course of a voting period, preventing them from proving that they voted in a given way via a zk-proof as that proof may not be based on the final vote. However, as the juror votes in Kleros must eventually become public to allow for redistribution of payoffs, even using this structure an attacker could create a smart contract with a deposit that would be burnt if she is unable to provide it with a traceable ring signature based zk-proof after the end of the case that a given vote that she controlled voted for Alice. If, in addition to having an anti-pre-revelation court, jurors were allowed to resubmit different commitments to potentially different votes, with the last submitted commit ultimately being the

        - The vote counts regardless, so this does not delay the main dispute
        - If this challenge is successful, the challenger receives the beta*minstake.
        - The supposed pre-revealer's side of a dispute can be crowdfunded. If the challenge is rejected, the supposed pre-revealer as well as her crowdfunders receive back a stake from the deposit of the challenger.
    - Activate commit+reveal, at least in the general court. Ideally, one would add a mechanism for turning this on in other courts later, particularly as we expect the two-step reveal process to become less onerous over the lifetime of this contract. If this is still not possible, err on the side of activating commit+reveal in many of existing courts/get community feedback on which courts to activate it in.
- Option in court parameters for whether to only draw jurors from PoH list so that each address gets at most one vote per case.
    - In particular, ideally if not too much additional complexity is required, addresses that are drawn in a previous round would not be eligible to be drawn in a subsequent round.
    - Let gamma be a new parameter that takes positive, possibly decimal values. Then each juror, who has not been previous drawn, should have a chance of being drawn of $\frac{(\#tokens\ staked\ in\ court)^{gamma}}{\sum_i (\#tokens\ staked\ in\ court\ by\ juror\ i)^{gamma}}$
    - Note would be complicated to recalibrate odds after each draw to eliminate address from pool, so instead maybe better to try to make further draws and check if draw new address. If one goes enough draws without getting new addresses, give up and have fewer juror spots than one would normally have in this round.
    - Ideally, the address of the PoH list against which the contract checks could be upgraded by the Kleros governance. In particular, this would allow one to swap out for the anonymized, ring-signature based version of PoH when it is available without having to launch a new version of the court.
    - Ideally, this feature could be turned on or off for a given court (so that it can be activated for a given court when one thinks that a sufficient population of qualified jurors for that court is present/registered on PoH) and/or PoH list could be changed in a way that isn't too disruptive. This depends on how the screen is applied, e.g. whether non-PoH addresses can be staked and just aren't drawn versus a system where the screen is enforced at the time of staking.

More minor tweaks to included if viable

- If the juror vote is sufficiently lopsided in a given round (so that it is highly statistically significant based on the sample of that jury that any other jury drawn from the same

---

only one which can be revealed/counted, this heuristically (I don't currently have a clean proof of this) would limit the category of attack discussed above to only those where the attacker locks up funds in a smart contract if she cannot ultimately provide a zk-proof that she voted in a given way. To the degree that we are worried about these types of attacks, the friction of the added capital lockup would probably reduce their frequency to the point of being less problematic.

subcourt would rule the same way) then appeal could jump directly to the next court, even if the number of jurors hasn't yet hit jurorsForCourtJump.[6]
    - Namely, check if alternative X has more than half of the votes in a round, check if $Prob(result\ or\ more\ extreme | true\ probability\ token\ ranks\ X\ first\ < 1/2) < \sum_{i=\#votes\ X}^{\#total\ votes} \#total\ votes\ C\ i\ (\frac{1}{2})^{\#total\ votes} < tuneable\ parameter$
    - Possible to use tail-bound on binomial distribution rather than computing this sum involving binomial coefficients if that is simpler. Also possible to code in specific thresholds for percentage/number of votes to be adjusted via governance.
    - Be careful about how used in general court (don't want to jump to a fork vote because some general court decision was highly unanimous and the contract thinks that the case should skip to the next level, i.e. forking - to the degree that the tuneable parameter is set by subcourt may just set equal to 0 for general court)
- Change how no juror coherent rounds are handled:
    - If there are votes, but no one is coherent with final ruling, take fees and lost deposits from that round and periodically airdrop them to addresses staked in the corresponding subcourt.
    - (Minor point - non-essential, but if possible, take one of these suggestions in a way that doesn't excessively complicate code): Special case for if there is a voting round where no one votes at all. Possibilities include:
        - Put fees towards eventual appeal fees
        - Allow a redrawing of the dispute with new jurors paid from these existing appeal fees
        - (The ideal here might be to return fees to the arbitrable contract, but this would require that arbitrable contracts have logic on how to process these refunds, so that is probably not worthwhile)
- Allow PoH listed identities to rate evidence (thumbs up/thumbs down) giving a rudimentary reputation scheme to address spam third party evidence. Might only be in UI rather than in smart contract if doing so is significantly simpler.
    - List party provided evidence highly/visibly regardless of score.
- Allow some kind of backup mechanism to submit evidence and arguments to the arbitrator contract/directly through the court interface (ideally in addition to being able to submit evidence through an arbitrable application) to avoid situations where an arbitrable application is poorly implemented and doesn't have this feature preventing evidence submission

[6] It is hard to do this precisely for cases where more than two alternatives have a significant number of votes when there is a non-trivial voting system. An alternative may have support that is highly statistically significant, and would essentially guarantee that any future sample from the subcourt would also win taking into account the subtleties of the voting system, even without getting a majority of the votes in a round. However, all of the voting systems we are considering have the "Majority" property, that if a given alternative is ranked first by a majority of votes, it wins, so we can at least check if there is an alternative with overwhelming majority support, avoiding "unnecessary" subcourt appeals in cases that are defacto binary even if we can't always avoid "unnecessary" subcourt appeals in cases that have multiple strong choices.

- Adjust stake rather than destake jurors who are incoherent and whose total PNK holdings drop below their stake.

Minor points that arise less out of recent research than being common sense add-ons/previously known improvements that were deprioritized in previous version

- Courts where juror fees are in tokens, particularly stable coins
- Liquid delegation in governance votes
- In order to avoid situation like that of Augur v1 where old cases resolve improperly because everyone migrated - have some mechanism in next court version to avoid this issue for future upgrades by allowing court governance to point court (as a sort of arbitrable app) to a subsequent version and allow some sort of appeal into the general court of the subsequent version? - edit: Clément indicates on team call that this should already be possible in some form for current court

Issues with Optimism:

- No access to blockhash

Potential solutions:

- VDF
- Chainlink
- API3?
- Signing from a set of actors (slash unresponsive actors)
- Check if Starkware VDF will be available on Optimism