cd smart

     npm i

     add .env file with seed phrase

     npx hardhat node

     >new terminal

     cd smart

cd ../dapp

npm i

npm run start

**Odds restrictions**

* one odds number

Standard odds are presented as a pair, with a spread so that simultaneous bets on both teams loses money for the bettor and makes money for the house. An obvious attack surface for a smart contract would be for the odds to imply an arbitrage as the offsetting bets would minimize the LP capital requirement, enabling the hacker to drain virtually all of the LPs capital at settlement. By using a single number that attack is eliminated. The 4.5% vig creates a competitive two-sided offer, a standard requirement for market makers on centralized exchanges.

The odds for the opponent are calculated via an algorithm. By restricting the odds to apply to the favorite, we can restrict the range of allowable odds, as no favorite has decimal odds greater than 2.000. This makes it easier to exclude bogus odds, as otherwise a hack might be to move odds from 1.500 to 4.00, which would enable a large payout. Updates are allowed to drift outside of the initial restnriction from (1.150, 2.000), to (1.075, 2.200), but that is still much less than if odds were submitted on both favorites and underdogs.

* outcomes only win-lose

Standard centralized sportsbooks cover diverse events on most days of the week, including exotic bets that are are not straightforward to validate. This demands a great amount of attention and competence by the oracle, and increases the probability that a minority of token holders take advantage of inattentive oracle token holders. The weekly reporting also makes the oracle easier to validate historically, in that the event logs refer to who won weekend events, which is easier to verify.

One could use a point spread, but that would not translate to MMA. Football, boxing and MMA will be the primary focus. The matches and odds are well-publicized early in the week. If there were a high profile events other than football and MMA can be accomodated on a case by case basis (for example, a World Cup soccer match).

* No extreme odds

Matches with extreme underdogs (eg, 10-1) are attractive for hackers, as the generate the most revenue for the smallest amount of capital. Initial decimal odds on favorites greater than 7:1 are not accepted. Initial decimal odds for favorites must be greater than 1.150, or less than an 88% probability of a win, or a 7-1 probability of winning. Such matches will simply not be covered. This would eliminate about 5% of NFL games historically, but is common among college football and MMA.

**Oracle submission restrictions**

* weekly settlement.

Games are constrained to start between the next Friday at 19:00 GMT until Tuesday; settlement cannot occur until Monday. Thus it is impossible to generate two settlements within a week. The weekly periodicity also makes the oracle easier to validate historically, in that the event logs refer to who won weekend events, which is easier to evaluate the oracle.

* A same 12 hour assured window to evaluate data

Data are submitted between 0:00 and 3:00 GMT. Voting by other oracle depositors can only occur after 3:00 GMT, and the vote cannot be tabulated until 13:00 GMT. Only one set of odds and results are processed each week. Thus ever oracle depositor can be assured of a chance to evaluate a data submission. O

* maximum of one daily submission, with 12 hours to evaluate

Data are submitted between 0:00 and 3:00 GMT. Voting by other oracle depositors can only occur after 3:00 GMT, and the vote cannot be tabulated until 13:00 GMT. Only one set of odds and results are processed each week. Thus ever oracle depositor can be assured of a chance to evaluate a data submission.

As no healthy adult sleeps more than 10 hours a day, all token holders will be able to vote before the data submission is processed. Submissions are not allowed on Saturday and Sunday.

Standard centralized sportsbooks cover diverse events on most days of the week, including exotic bets that are are not straightforward to validate. This demands a great amount of attention and competence by the oracle, and increases the probability that a minority of token holders take advantage of inattentive oracle token holders.

* Maximimum 32 events

The settlement function loops through the events, and 32 is big enough to capture most weekend events. Gas is a constraint, but this has the added value of making the contract easier to monitor and validate, as obscure contests would be more difficult to asses.

This dapp consists of three solidity contracts: betting, oracle, and token. In contrast, Uniswap’s V3 ‘contract’ contains 31 contracts, which makes it difficult to audit. One can evaluate functions piecemeal, but with tens of interacting files, many upgradeable, it is difficult to span the state space configuration that may make an otherwise innocuous argument a problem. Users must take security on faith. With ASB, one can download the three contracts, and test different different scenarios to find a hacking surface. I provide a dozen hardhat tests as templates to build upon in my GitHub repo.

* Binary final outcomes only

Ambiguity is created by exotic bets that are difficult to evaluate, such as who scores first. These bets are only determined by their final outcome, which may include a point differential. All the oracle evaluators need are the results from the prior weekend.

**Simplifications**

* only three contracts

one a conventional ERC20 token. In contrast, many DEX contracts have dozens of contracts, which are then difficult to audit.

* contracts not upgradeable

Static contracts remove any need for governance to vote on upgrades. Most importantly, it means there is no group of developers managing, promoting and proposing changes. Such developers would need to be paid, and generally this requires a corporate structure. Such corporations are attack surfaces for censors. This also removes the risk from bugs often found in upgrades.

* No adjudication process

A protocol for disputing data validated by the oracle generates considerable delay. The oracle incentives are based on the present value of the oracle token, which should be sufficient. Redundant mechanisms lessen the incentive for the oracle token holders to monitor and discipline data submittors.

* Everything in AVAX, no stablecoin

By using native AVAX for all bets we eliminate unnecessary costly swapping into and out of stablecoins. As stablecoins are generally centralized, we eliminate an attack surface as well. Users will have to bear avax price risk, but this is a minor inconvenience relative to the extra costs created by requiring users to buy a stablecoin.

**Safety mechanisms**

* halting betting on matches

A match’s odds may become obviously stale. Allowing the oracle to turn such matches off does not expose the contract to malfeasance, it just prevents more bad trades from happening. These can be undone, as well.

* Unintentional error robustness

Oracle submitters can resubmit in the three-hour window they have to submit data. This is aimed at the case where a submission contains an obvious error that the submitter did not notice until it was posted. This still gives the oracle collective 10 hours to evaluate the data.

If an initial datasubmission is rejected, the week is not ruined. A replacement can be made the following day, allowing the contract to function that weekend. As it is irrational for the oracle to deliberately choose to cheat, unintentional errors are the main concern, and often these are not seen until one actually sends the data to the contract. By allowing a data sender to resend if they do so within the next few hours, it does not complicate anything for the other oracle token depositors evaluating this data.

There is no financial penatly for rejected submission, as we want to encourage the oracle members to reject data without fearing retaliation.

**Bookie**

* No bookie withdrawals during active betting

Once odds have been posted an betting is active, bookies may not like their exposure. More blatantly, if a result implies a big bookie loss, they would have an incentive to remove funds before settle. The contract prevents any bookie funding or withdrawals between the time odds are sent to the betting contract, and settlement. 1% fee charged for withdrawing within the same epoch as a deposit

Flash transactions enable efficient arbitrage, but the benefits here are low. A malicious bookie may spoof volume to discourage other bookies, and once they are scared away, remove the excessive liquidity. This discourages that tactic.

**Oracle restrictions**

* No token withdrawals during vote

prevents accounts from double voting

* Oracle has minimum and maximum token requirement.

Minimum requirement makes the cost of evaluating or sending data less than the expected loss from a reputation-destroying fraudulent data submission.

* LPs are only paid if they vote

This motivates the oracle token holders to evaluate the data

* LPs are charged a 1% fee if they withdraw after funding prior to settlement.

If LPs could deposit and withdraw quickly without a fee, some LPs might find it profitable to scare away other LPs by depositing large amounts merely to discourage other LPs, who would see low expected returns given a large capital base. Once scared away, the malicious LP would withdraw capital to make the return attractive. This tactic does not help the LP collective.

* Oracle submitters must differ sequentially

This motivates oracle memmbers to create their own data submissions, as they cannot depend on a single oracle member who appears to dominate data submissions.While it could devolve into two oracle accounts posting all of the data, it is a small nudge in the right direction. If a token holder creates a data submission, but finds they did not post in time, they will be prepared to give a good evaluation of the data submitted.

**Betting restrictions**

* Min bet size 1 avax

While testing with trace amounts can be fun, there are DDOS-type of risks generated by allowing trace-amount transactions.

* No betting after game start time

Obviously, a match with fixed odds is easier to bet on when it is partially completed. The ambiguity of blocktime is insignificant to this requirement, as a mere couple of seconds would not expose the contract to risk

* bets constrained by bookie capital
  + ex ante limit of LPcapital/x per match, where x is a number chosen by the oracle. For example, if x=10 and total capital was 100, then the LPs would have at most 10 avax at risk to any one particular event.
  + If 10 units were available, and the concentration parameter was set to 5, then five matches could have 2 units of exposure to the LPs, and use up all of the LP capital. A further bet of 1 unit would not be possible because there would be no more free LP capital left.

The front-end is helpful, but not necessary. One can use Remix, python, or any other method of transacting with the contract, which is permissionless.

## Odds in the contract

Odds are available on many betting websites, and arbitrage limits how far these odds can differ. On average, a team's implied probability of winning will change by only 2% over the week, rarely over 5%. All odd postings and updates are recorded in event logs, observable in online queries at sportAVAX.co.

A contest will have a single odds number posted for a contest. These odds are supplied only for the initial favorite using a truncation of preliminary decimal odds. For example, 1.909 would be stored as 909, 2.50 as 1500, etc. This number, however, is just relevant to the team in slot 0, the initial favorite. Further, it needs to be adjusted to reflect the oracle fee that would be assessed to the winner payout. Thus, the betting odds for a favorite where the match odds were 957 would be 909, via

Net Odds (favorite) = (contractMatchOdds \* 0.95)/1000 + 1

= 957\*0.95/1000+1=1.909

The gross odds for its opponent are generated within the contract by the following formula:



Then to account for the oracle take, the all-in odds for team 1 would be

Net Bettor Odds (underdog) = (underdogOdds \* 0.95)/1000 + 1

With this method, we can ensure that the set of odds for a contest generates a positive vig, removing a potential attack vector.[[1]](#footnote-1) This formula generates a vig of 2.5% for the LPs via parameter 45 in the above equation, and the 5% take of winnings generates an approximate 2.5% vig for the oracle.

The standard odds presented by most casinos embody the standard vig of 4.5%. For example, -110 on an even-money bet, which is 1.909 in decimal odds. ASB applies the vig to each contest via an algorithm that works well, though it is simply a hack as opposed to something derived from axioms. The approach is to put the LP’s take into the ‘spread’ between the odds offered for the favorite and the underdog, but leave the Oracle’s take out of the spread. The Oracle fee is then taken out of the winnings.

Initial odds presented must be between 1.999 and 1.125. The cap at 2.000 reflects the fact the initial odds apply to the initial favorite, while the 1.125 minimum removes events where the initial odds are greater than 8:1, lopsided contests. Eliminating high payout contests mitigates risk, as such events would invite hacker attention.

The excel spreadsheet is provided that generates the data in the necessary format. The basic algorithm is this

1. Take an initial set of odds
   * Home team: +135
   * Away team: -150
2. rearrange so that the favorite team is first
   1. team[0]: -150, team[1]: +135
3. Translate into win probability
   1. team[0]: 59.8%, team[1]: 42.4%
4. calculate probability spread
   1. spread = 0.598 – 0.424 = 0.174
5. Calculate new favorite, team[0], prob(win)
   1. prob(team[0] win) = 51.1% + spread/2
   2. prob(team[0] win) = 51.1% + 8.7% = 59.8%
6. Translate prob(team[0] win) into decimal odds.
   1. decOdds(team[0]) = 1 / 0.598 = 1.6716
7. Translate decimal into the payoff of the bet. This number will represent the match odds in the contract.
   1. contractOdds = 1000 \* (decOdds – 1)
   2. contractOdds = 671
8. Translate into netDecimalOdds(team[0]) presented to bettor
   1. 0.95\*671/1000 + 1 = 1.637
   2. The oracle fee reduces the actual decimal odds returns presented within the contract
9. Translate into team[1] payoff
   1. contractOdds(team[1])= 1e6 / (671 + 45) - 45 = 1359
   2. net decOdds (team[1]) = 0.95\*1359/1000 + 1 = 2.291

In this example, the vig is 4.5%: 1.637 \* 2.291 / (2.291 + 1.637). The above algorithm generates a vig near 4.5% across the range of odds covered in this contract.

The website avaxsportsbook.com displays the decimal odds users receive if they win. For example, a user seeing odds of 1.900 will receive back 1.900 times their bet amount.

## Redeeming a Bet

Bets are stored in a mapping within a better’s struct, and after 10 bets, no further bets can be made until they are redeemed. Redemptions can only occur when a bettor has no active bets, so a bettor should redeem his bets after settlement if he anticipates a problem. All bets in the array are settled for the bettor. Each bet is represented by the unique combination of epoch, match, and pick. At settlement, a bets hash refers to a struct containing this information, and a mapping generated at settlement allows redemption. Users to redeem bets by clicking a single button (it is one transaction).

## LP Revenue

LPs own a pro-rata portion of the contract's revenue based on their percentage of LP capital before that week's events. Statistically, the LP capital will grow each settlement due to the vig; this is how LPs make money. As the relevant LP credit/debit occurs at settlement, the LP's AVAX/share value is fixed each week when users can withdraw or invest.

An initial investment generates the following shares:

LPshares = AVAX invested × TotalLpShares / TotalLpAvax

For example, assume the contract has 123 AVAX owned by its LPs, who have 100 shares. This AVAX may be sitting free or locked up as collateral for upcoming contests. This implies each LP share is worth 1.23 AVAX.

LP AVAX LP TotalShares avax/Share

123 100 1.23

Suppose Alice wishes to invest 10 AVAX into this pool. The above formula implies she would receive 8.13 shares (10/1.23). This would change the pool's balance sheet to

LP AVAX LP TotalShares avax/Share

133 108.13 1.23

Note the ratio of AVAX/share is the same after Alice's investment, so existing shareholders do not lose or gain money via Alice's new investment.

If we assume the LP collective gained 2 AVAX that week, the new balance sheet after a settlement will look like this:

LP AVAX LP TotalShares avax/Share

135 108.13 1.25

The increase from 133 to 135 reflects a 1.5% profit from that epoch's games. If Alice then sold her shares, she would receive AVAX using a transformation of the above formula:

avax Withdrawal = TotalLpAvax × SharesSold / TotalLpShares

Selling 8.13 shares would generate 10.15 AVAX, a 1.5% return on their investment, identical to how much the AVAX LP pool rose over that period.

In this way, any LP investment or withdrawal reflects the percent change in the size of the LP pool’s avax/share over the investment period.

## Oracle Revenue

Oracle token holders must deposit their tokens in the oracle contract to vote, and they must vote to receive revenue. When a weekly settlement transaction is executed, the oracle's 5% fee is applied to the winnings and sent to the oracle contract. The '*feePool'* state variable reflects the lifetime amount of AVAX per token paid to the oracle contract.



When an oracle token holder deposits into the contract, their account notes the current value of *feePool*. When that oracle token holder withdraws or adds to their account, the token holder is sent their entire accrued AVAX using the formula



Having tokens in the oracle is a necessary but insufficient condition for being paid. The contract then takes the total number of tokens





This account's *OraclePoughback* is sent to the Oracle *feePool* as if it were revenue from a settlement.

There is no scenario where the token holders can lose accrued revenue, either due to a lucky win streak by bettors or an oracle hack. Token holders can be sure the contract is in balance, where accounts payable are equal to AVAX in the contract at all times.

**How to bet**

only when betting is active (odds have been sent to the contract by the oracle)

only prior to the game start time (sent in schedule using GMT UTC)

constrained by LP capital

click

type in amount of avax on bet

click bet

**How to redeem**

only when there are no active bets

after 16 bets, user must redeem so account can bet again

One can redeem whenever. Funds do not disappear if never claimed.

wait until settlement, which happens early in the week, to remove active bets

click redeem

**How to fund betting account**

only when betting is not active

betpage

fund: enter amount and click fund

**How to invest in the house**

only when betting is not active

go to bookie page

fund: enter amount and click fund

**How to deposit tokens in oracle**

approve:

Deposit:

**How to withdraw as bettor**

withdraw

**How to withdraw as bookie**

sell ‘shares’.

1. A negative vig would allow someone to create positions that would generate arbitrage profits. [↑](#footnote-ref-1)