AvaxSportsBook

Decentralized Sports Betting Contract

Eric Falkenstein

11/24/23

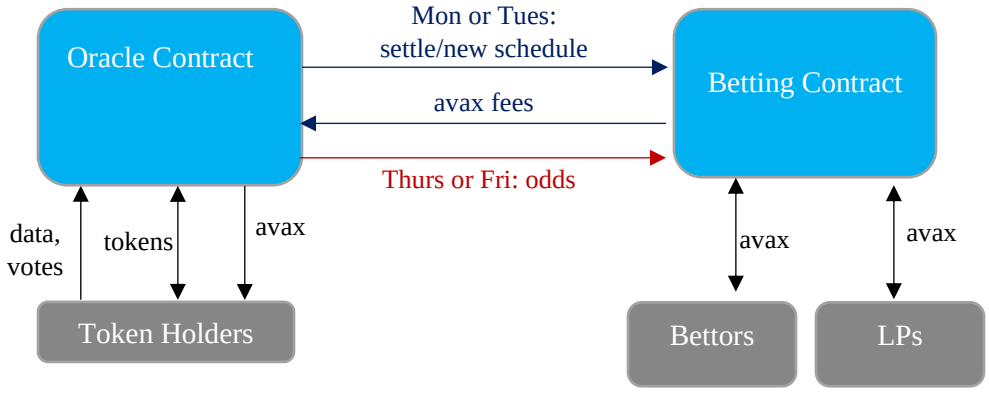
Abstract. AvaxSportsBook is a blockchain smart contract for straight-up betting on weekend sporting events. The unique stability of sports odds relative to the spread makes it a perfect application for a digital vending machine. Users can bet or join the house book, which provides liquidity for residual imbalances. Cross-margining allows a finite amount of LP capital to support unlimited bets. The standard 4.5% vig is applied to each contest and splits evenly between the liquidity providers and its unique oracle, which provides the weekly slate of events, odds, and outcomes. The contract is completely on-chain, immutable, permissionless, transparent, and decentralized, providing a convenient way to bet on prominent sporting events.

Introduction

Sports betting is ideally suited for a completely on-chain smart contract. The peculiar nature of sports odds—their stability and implicit bid-ask spread—eliminates most of the need for low latency administration.[1] People do not need better odds so much as easy access to ubiquitous conventional odds. No 'wisdom of the crowd' is needed to add efficiency to any big sportsbook's odds, making the oracle's consensus mechanism a simple exercise in filtering out blatantly wrong data.[2] It is straightforward to apply escrow accounting logic where bets are cross-margined for the house, permissionless, and secure.

A focused sports betting smart contract can provide a service big enough to matter but small enough to manage. It focuses on 32 weekend events and requires only two weekly data submissions, making it feasible for individuals with limited time and money to administer. On Avalanche, the bettor transactions are under 10 cents, making common bets of $20 or $50 economically attractive. Everything is open source and users can access the contract via the front end in an archived GitHub repo available via Vercel, Spheron, a server at www.avaxsportsbook.io, or download the front end and run it locally. Users can also download a gist of contracts into Remix (here).

There are three types of *AvaxSportsBook* (hereafter, **ASB**) contract users: bettors, liquidity providers, and the oracle collective. The total vig is split evenly between the oracle and the LPs.[3] Bettors can take either side of any regular bet, subject to a bet size constraint based on the amount of free liquidity provider (LP) capital. An oracle token holder submits last weekend's results and next weekend's slate of matches and start times early in the week (e.g., Tuesday). Odds are posted late in the week (e.g., Friday), after which bettors can bet up to game time on the various events. Bettors can redeem their bets immediately after settlement, and LPs can withdraw or fund only when betting is inactive (from when odds are posted until outcomes are posted and bets are settled, a few days each week).



LP exposures are automatically cross-margined to minimize required capital. For example, consider an even bet with no fee, and the decimal odds are 2.00. A house book with 10 AVAX collateralizes a bet of

[1] Adverse selection for LPs would be when they offer odds and get filled only when the trade makes money for the trader, implying the LP loses money.
[2] The wisdom of experts, and the masses, are already in a sportsbook's odd.
[3] It is equal statistically, that is, over time. Both the oracle and LPs are subject to small sample variation from different sources. Some weeks there may be no winners, so the oracle gets nothing; some weeks the LPs could lose all bets where they have net exposure. The oracle must work, the LPs must take risk.

10 AVAX, and also when the book has 510 on one team and 500 on its opponent.[4] LP capital of *X* AVAX supports a book of infinite size when bets of size *X* are made sequentially. The required LP capital on any single event is a function of the maximum *net* payout, which is the payoff on a team *minus* the amount bet on its opponent. An adjustable parameter limits how much of this capital can be applied to any single event, limiting LP exposure to any single event. For example, if LP capital is 100 AVAX, a parameter of 10 restricts the net LP liability for any one contest to 10 AVAX.

Legal

The main problem in developing such a contract is monetizing it isn't easy. Administering, funding, or owning an unpermissioned sports betting contract is illegal in many regions, such as the USA, where I live. However, the blockchain is global, and millions, perhaps billions of people worldwide, can legally act as liquidity providers, oracle administrators, and bettors.

I created a blockchain app that works, and abandoning the hope of profiting from this contract is better than never releasing it. I have no control or financial interest and cannot turn off the contracts. There is no ASB foundation, ICO, or governance issues regarding upgrades. While I may comment on this contract on various platforms, I will not generate any more code (my GitHub repo is read-only and archived).

Three independent initial oracle accounts administer the contract. As I worked on this for several years, I eventually found three who would act as oracle administrators. I provide tools to make their job as easy as possible, which are available to everyone in the GitHub repo. They are whitelisted for adjusting the concentration factor that controls diversification and halting bets on specific contests.[5] I am confident these initial three will act in the best interest of the contest, and these minor powers protect the contract, especially in the early stages when errors may arise unintentionally. I will stop all communication with them when this is publicly announced, and they will probably, at least initially, read my tweets and substack.

Sixty percent of the oracle tokens are distributed via an initial LP rewards program. These tokens have already been minted and are sitting in the betting contract. No mechanism exists to create more of these tokens (1 million).

Oracle Incentive Compatibility

Given accurate odds and results, LPs and bettors cannot cheat as the contract's internal accounting applies that information to the LPs and bettors. ASB's oracle can cheat, so the key is creating an incentive-compatible contract, where honesty is the oracle's profit-maximizing tactic. It is *trustless* because one must only trust the oracle is financially self-interested, an objective shared by good and bad people. The key to this is simplicity. By restricting the dapp's focus, we minimize the state space of user actions, contract data, and the interpretation of these data.

Sports betting is a competitive market, so the standard 4.5% vig reflects an equilibrium balancing the demands of bettors and bookies instead of monopoly power. By taking the standard vig as a given, we remove naive schemes that, in theory, are less costly but, in practice, create failed markets (e.g., Augur). The hassle-free ability to bet or be the house should be sufficient to make it a dominant alternative for

[4] A flat book does not mean the bet amounts for both teams are equal. For example, a contest could have 5 avax bet on a 5:1 favorite, and 1 avax bet on the underdog, leaving the LPs with zero risk.

[5] Regular oracle depositors are limited to one concentration factor or event halt per epoch, and then cannot withdraw until the end of the epoch. This prevents a rogue oracle account from effectively shutting down the contract by halting all the matches or post a stupid concentration factor.

many sports bettors. If bettors find the contract's vig too generous, they can join the house and receive income proportional to the liquidity they provide.

Creating a game where honesty is an oracle's best strategy is straightforward. As long as the potential cheat gain is less than the present value of future revenue foregone, it will be in the oracle's self-interest to provide accurate data. Incentive compatibility is critical to low-cost enforcement of contracts.

Consider the following cost-benefit analysis for ASB's oracle. Assume a betting contract has 100 in net exposure, which we will conservatively assume is the book's gross betting exposure (no offsetting bets that generate reward but no risk to the LPs). As the oracle's fee is half the vig, this would average about 2.5 AVAX weekly revenue. Given 50 settlement events over the year, this annualizes to 125 AVAX. Given a conservative 10 price/earnings ratio, this values the oracle collective at 1,250 AVAX. This example's maximum potential cheating revenue is 100 AVAX, so the LPs have net and gross exposure to the wrong side of every bet made by the cheating oracle's sock-puppet bettor. A voting majority's oracle tokens have a present value of 625 AVAX in the above example, significantly more than the 100 AVAX in a cheat.[6]

Such a scam would be conspicuous in the readable, weekly event logs showing what games, odds, and outcomes were reported. Unlike oracles that service many contracts, the ASB's oracle collective cannot assert the cheat was merely an unrepresentative anomaly. Each week, they are tasked with producing two slates of data, where a single bad data point—one beyond the standard variance in odds across sportsbooks, a late start time, or an incorrect outcome—taints the entire slate. The oracle collective is 'all in' on this one betting contract that cannot be upgraded or extended.[7]

Incenting the oracle properly protects the contract against insiders, the current token holders. In contrast, decentralization defends this contract against outsiders. Powerful institutions have always used centralized power to prevent competition, often using disingenuous rationales emphasizing safety. They need a choke point, prevented if a collective of pseudonymous accounts worldwide administers the oracle and provides liquidity. The fungibility of oracle tokens allows an oracle suspicious of attack to sell his position and move on. The oracle's duties are simple enough for most people to master.

I provide an Excel sheet and MS Access database for creating the data in the format required by the contract and sending transactions to the contract via Python. Once created, one can create simple cronjobs that then post data at the correct time. The objective was to make administering the contract feasible for a student to do this as a hobby, taking only an hour or so each week.

Oracle Vaults

Oracle token holders only earn dividends if they vote on data submissions.[8] For example, if a token holder voted on 50% of the data proposals, she would receive half of her potential revenue. However, the cost of a meaningful data evaluation is relatively fixed, comprising mainly time and attention. The rational strategy for a small token holder would be to automate an uninformed vote to ensure 100% of her potential revenue, but this willful ignorance creates an attack surface. To minimize this tendency, all token voters need at least 10% of the lifetime supply to deposit and vote on the oracle contract. Larger

[6] 625 = 50.001% of 1250

[7] Augur was plagued by bad-faith actors like Poyo-Poyo, whose intentionally deceptive bets were dismissed as the actions of a rogue agent. If the implications of his actions were immediately fatal for Augur he would have been disciplined by those with an equity interest in the dapp.

[8] A proposal attributes the submitters tokens as a yes vote for that submission.

token holders will bear significant costs if one of the other token holders submits fraudulent data, justifying the time needed to evaluate or create oracle data.

The token floor incentivizes smaller token holders to create or join a vault contract where the administrator can charge a palatable fee for his services because he amortizes his time and effort costs across many token holders. There will be mutual gains of trade for both sides: the small oracle token holders and the administrators of vaults. It targets a representative versus direct democracy, a consequence of Robert Michel's *Iron Law of Oligarchy*.[9]

The vaults should not be too big, as this would present an attack surface for censors and hackers. Thus, each token deposit account in the oracle contract is capped at 220 thousand tokens (ie, 22%). The vaults should be independent, making the contract more robust, and maximizing the present value of their tokens.[10] If the contract works, others will be incentivized to create vaults, and if it is not popular, I would have wasted time creating them.

Timing of Oracle Data and Betting

Each week, the MMA, boxing, and US football games that weekend are sent to the oracle. These events are highlighted because they have odds early in the week. Sports like basketball and baseball play several times a week, and odds are often not presented for games after the most immediate contest. I do not follow baseball, hockey, or soccer, so I hesitate to apply this contract to those sports. However, someone who understands these sports can extend the contract presented to handle them reasonably.

Each game is given a starting time between next Friday at 9 PM GMT and Tuesday at 9 PM GMT. When the betting contract has odds data for the following weekend, bettors can bet up to the time of the games that weekend. LPs cannot withdraw or deposit from when odds are posted (~Friday) through a settlement (~Tuesday), as otherwise, LPs could game the contract by anticipating unusual losses or winnings as they accrue over the weekend's events.

After the weekend, the outcomes are sent to the betting contract bettors, which settle that week's bets. At that point, LPs can withdraw, and bettors can redeem their bets. The contract cannot seize neglected funds, so as long as the blockchain exists, users can safely let unredeemed money sit in the contract.

The data submission process will look like this:

Day	GMT (London)	What
Wednesday	midnight – 3 AM	outcomes/new schedule sent for evaluation
Wednesday	3 PM	outcomes/new schedule sent to betting contract
Friday	midnight – 3 AM	odds sent for evaluation
Friday	3 PM	odds sent to betting contract

[9] In *Political Parties* (1911), Michel noted that that rule by an elite, or oligarchy, is inevitable within any democratic organization as part of the tactical necessities of the organization.

[10] Vault independence is not enforced in this dapp, as all depositor accounts could be in the same wallet. The vault creators should know it is in their best interest to be independent. They would be worth more, collectively.

Settlements should happen Tuesday evening, though if there are no Monday night events, it could happen Monday, and if a data submission is rejected on a Tuesday, it would be resubmitted the next day.

Posting data can only occur in the three-hour-minute window where the hour is between midnight and 3:00 AM GMT, which is 8-11 PM in the Caribbean, Venezuela, and Paraguay, and 8-11 AM in Brunei. The initial data sender can costlessly send corrective submissions that replace their earlier ones until 3:00 AM GMT. This anticipates accidental fat-finger mistakes that are only obvious once on the blockchain. Voters vote on the last dataset submitted between 3:00 AM and 15:00 GMT, after which anyone can execute the function that tallies the oracle vote. A majority yes vote sends the data to the betting contract; a failure pushes the data submission back a day.

Settlement data cannot be sent on Saturday or Sunday. Oracle administrators do not need 24/7 vigilance. The aim is that a single person with student-level resources of time and money can fully attend the contract's oracle.

How Oracle Token Holders Claim Oracle's Revenue

The oracle accrues fee revenue at weekly settlement. Applying a 5% fee to the winnings generates a 2.5% take, about half of the contract's fee revenue. For example, if there is one bet of 1 AVAX with gross decimal odds of 1.957, the bettor gets back his 1 AVAX and then 95% of the 0.957 AVAX generated by his win (~0.909). Note this implies the oracle token depositors have no economic risk, unlike the LPs.

Another way the oracle token holders can accrue revenue is when negligent token depositors claim revenue. If an oracle account does not vote each time, their accrued revenue is slashed by the percentage of votes they missed, and the lost amount is reallocated to the other token holders. For example, if a token deposit account spanned three settlements and six data submissions and voted three times, they would receive one-half of their payment. When this account tries to claim their oracle revenue, the forsaken half would be added back to the pool, going to the other token holders.

Betting and Redeeming

Bettors redeem all their outstanding bets in a batch. The redeem function loops through up to 16 bets in a user's account and credits any winnings to the bettor's account. Redemption can only be processed if there are no active bets in the account, so bettors must wait until the subsequent settlement to redeem if they have any active bets. If an account has 16 unredeemed bets, it must redeem them before it can place another bet. All ties and 'no contest' games return bettors their initial bet. Winners receive their bet amount plus the payoff implied by their bet odds.

Bets are stored in a mapping within a better's struct, and after 16 bets, no further bets can be made until they are redeemed. 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. Redemptions can only occur when a bettor has no active bets, so a bettor should redeem his bets after settlement and before betting again if he wants to redeem.

Liquidity Providers (LPs)

The LP's total capital is available equally to all contests that week, but there is a limiting mechanism on how much AVAX can be allocated to any contest. Specifically, the betting contract parameter concFactor (concentration factor) is used so that given LP capital *x* and concFactor *y*, the maximum exposure to any match would be x/y. This diversifies the LP's risk, eliminating the chance that a single contest outcome could extinguish LP capital. The betting contract contains all the methods for bettors and LPs.

LP capital backstops residual imbalances in the book, and given the finite number of events, the LP can lose money over a weekend, though this becomes statistically insignificant over time. However, the LP's main risk is the black swan risk via oracle fraud, something any good hacker would strenuously avoid until they cheat. A hack would almost surely use a bettor-oracle conspiracy to maximize their cheat payoff, as the LP exposure is passive, while a bettor can maximize the net payout. Thus, LPs are incentivized to become oracle token holders to align their incentives, as the most likely cheat would involve a conspiracy between the oracle and a bettor, defrauding the passive LPs. The token rewards allocation to the initial LPs encourages an LP/oracle overlap.

To become an LP, one sends AVAX to the betting contract, which then credits the LP with shares representing their pro-rata ownership of the LP pool. For example, if there are 10 AVAX in the LP book and 10 shares, adding 1 AVAX would give a new LP a $1/11^{th}$ share of the new pool of 11 AVAX, keeping the AVAX/share value the same. This LP claim exists only within the betting contract, is not transferable to other AVAX addresses, and is not represented by a token. LP shares are like a stock at its net asset value; tokens are like a stock's market value.

The size of the LP capital should adjust to the volume and degree of cross-margined betting, the more of which increases the LP's expected return. For example, a book with 10 bet on team A and zero on its opponent will generate an expected return for the LP, statistically, in that over time, the vig in the odds spread implies bettors need a 2.2% edge in predicting winners to beat the house, which is difficult (as proven by the nice casinos). If a book had 110 bet on team A and 100 on its opponent, the required LP capital would be the same, but here the LP would make a riskless return on the offsetting bets in addition to the risky 10 AVAX (which would have an expected return of zero, either +10 or -10). In this example, the gross betting exposure in the latter case is 21 times larger than the net exposure. The greater the gross-to-net exposure ratio, the greater the LP return.

Not only is the net/gross betting data unknown, but the max(net-to-gross) ratio is important because limited LP capital will prevent bets, and potential bettors might not return later if bettors take the other side, allowing their bet. Thus, although a finite amount of LP capital can support unlimited betting, too little will lower the LP's return on capital if casual bettors attempt to bet when the contract is maxed out on their bet of choice. Conversely, too much LP capital would also lower the LP's return. The average net/gross ratio will be revealed over time and affect the return for a given level of LP capital. It will be an empirical issue what the correct amount of LP capital is for an expected gross betting amount.

Cross-margining LP exposure

The contract's logic makes sure all bets are fully collateralized. As bettors take the opposite side of a contest, it is a waste of capital to require the LPs to collateralize both sides independently. The solution involves netting exposure upon every bet.

In betting on a limited number of binary outcomes, the worst-case scenario for the house is assumed (i.e., each LP net position loses). The contract will always be fully collateralized on all bets, as this is enforced at the time of each bet. Adjusting the net required LP margin involves 'linear programming' where the LP's net game exposure is the maximum LP liability for each contest. The margin adjustment is applied at the time of a bet, so there must be sufficient free LP collateral to accommodate a bet, adding to LP exposure.

For example, assume two teams are given even odds so that for either team, a 1 AVAX bet pays the winner 2 AVAX. If there are 10 AVAX on team A and 10 AVAX on its opponent, team B, it would be a 'flat' book in that the LPs have no exposure to this game; payoffs are funded by betting counterparties, not

the LPs. A new bet that pushes the book to have a net exposure would necessitate LP funds as collateral, so a bet of 2 AVAX on team A would add 2 AVAX to the LP's locked Margin. Given a total of 12 on team A, and 10 on team B, a bet of 2 AVAX on team B would move 2 AVAX *out* of the locked Margin because the book would be flat again.

LP Oracle Token Rewards

50% of the maximum oracle token supply, 500k, was sent to the betting contract as a reward for initial LPs. This encourages an LP/oracle overlap, which helps align incentives between the oracle and the LPs. Each week, 30k tokens are available for reward distribution, and each week, LPs can send a function to receive their pro-rata share (e.g., an LP with 10% of the shares would receive 3k tokens). Rewards do not start until epoch 3 to avoid the initial oracle token holders accumulating most tokens while the contract has little visibility. The incentive program will last until all the tokens are distributed. Minting more than the initial one million tokens minted is impossible.

Avalanche

I used to be an Ethereum maxi, but I became convinced that Avalanche's blockchain is significantly more efficient.[11] Avalanche's Proof-of-Stake consensus mechanism dominates Ethereum's because its validators survey a fixed number of other validators as the network grows. It should come as no surprise that a second-generation consensus mechanism would generate a significant improvement from the initial Nakamoto consensus outlined in the Bitcoin white paper.

Ethereum is stuck with an inefficient consensus mechanism because its L2s are now too powerful, which reduces the incentive for making the main chain more efficient, as this would make the L2s unnecessary. The bottom line is that Avalanche is as cheap and fast as an Ethereum Layer 2 blockchain but with greater decentralization.

I call it *Avax*SportsBook to remind people which chain they must put crypto on. Avalanche uses the same contracting logic as Ethereum, so it took no extra work to port my Solidity and Web3 files onto Avalanche's C-chain. Avalanche has the same address structure as Ethereum, so users can use their MetaMask wallets to store and transact with on Avalanche's C-chain. Avalanche's Core Wallet (or core.app) provides a secure bridging mechanism to move Bitcoin or ETH onto Avalanche's chain as wrapped tokens, which can then be swapped into AVAX via on-chain AMMs. I recommend using the Core wallet just for bridging and then MetaMask for transactions (that is my preference; to each his own).

GitHub Repo

The SDK is available in an archived (static read-only) GitHub repository that others are free to copy and extend; the front end is available on various websites, but this can be downloaded from GitHub to one's desktop. There are tools for pulling the event log data to assess the oracle's credibility. These are readable logs that do not require obscure knowledge about hashing functions.

I encourage people to copy or extend the contract. My incentive is to create something popular because there is joy in creating something people like, and I want to encourage crypto use in general. If it fails, all the time and effort I invested in this project would be a waste, so I am invested in it even though I have no financial interest. In contrast, if someone merely copies my code and rebrands it, they have little to lose if it turns out their code has a backdoor enabling a scam. Users should be wary of copycats whose incentives generate a significantly higher probability of cheating, and this should generate a modest barrier to entry for blatant replications.

[11] The earlier version of this was call ‘sporteth,’ and that may sometimes appear in the code or documentation.

Audit

I think the contract is safe, but I realize I am fallible, and could benefit from a knowledgeable third-party review. However, as I have no financial interest in this dapp I did not pay for an audit. There are several test scripts in the repo, and one can build upon these to assess the contract for bugs or attack vectors. A casual bettor who does not have the ability to audit these contracts themselves should just post modest sums so a worst-case scenario would not be too painful.

Conclusion

Most sports betting sites touting their crypto functionality are just conventional online betting sites accepting crypto. A truly blockchain-based betting dapp upholds Satoshi's vision of *pseudonymity*, *confiscation-proofness*, and *permissionless access*, which requires it to have no off-chain presence. I hope that a focused dapp with proper incentives can provide an example of what blockchain smart contracts can do. The purpose of the contract is to facilitate casual sports betting, not create a new token for people to pump or a new protocol that can potentially replace both Amazon and Goldman Sachs. ASB's oracle token holders have a straightforward but essential job that generates instant revenue.

A sustainable contract creates a repeated game where honesty is always the dominant strategy for every player. The trust one puts into the ASB Oracle is fundamentally the same as why investors trust miners: the rational self-interested assessment that honesty dominates dishonesty. Paradoxically, adding multiple redundant adjudicators gives the illusion of greater safety while creating a state space for a hack that grows exponentially in the number of players and actions they can take. ASB's simplicity enables an incentive-compatible contract, allowing bettors to easily access conventional odds on big games and cash out quickly.

ASB is a straightforward application of escrow logic to an everyday use case. It provides an efficient way to bet on big games without the hassles presented by online betting services. Sports betting is ubiquitous, but it should be easier. This contract provides a simple way to do that.

Appendix

Game Theory Objectives

In the game theory field of *mechanism design*, two conditions exist for a sustainable contract. First, an *incentive compatibility* (IC) constraint motivates honest or cooperative actions by players. Incentive compatibility is vital to low-cost enforcement of contracts, and historically, this mechanism centered on reputation, not contract law administered by the state.[12] The blockchain's transparency, immutability, and pseudonymity make reputation much more accessible to monitor. When agents have incentives aligned with their counterparties, we minimize non-explicit costs like delay and spread that plague these markets.

If the bettors or LPs want to cheat, they need to collude with the oracle; if the oracle does not cheat, neither bettors nor LPs can cheat. Thus, we must ensure honest reporting is always the oracle's best strategy, per the IC constraint.

Honesty may be a player's best strategy conditional upon playing a game, but they might find not playing the game their best strategy. For example, if there is a fixed cost of $100 to play the game, and playing honestly generates $50 revenue vs cheating and making $40, the game would be incentive compatible conditional upon playing, but not playing is the rational choice. This highlights the second requirement for a sustainable game, the *participation* constraint. All necessary parties must find it beneficial to use the contract. ASB focuses on making the oracle job as easy as possible, as complexity generates costs, primarily in time. The GitHub repo provides tools for gathering data and Python files for sending oracle transactions to the contract, making the cost of being an oracle low.

Weekly settlement applied to an immutable betting schedule creates a repeated game. Repeated play is essential in moving the game-theoretic equilibrium from bad to good, as demonstrated by the different equilibrium for the prisoner's dilemma when it moves from a one-period game to a sequence of one-period games (see Robert Axelrod's *Evolution of Cooperation* (1982). Importantly, Axelrod's famous experiments involved a game with 200 iterations and consistency in several dimensions—same choice, equal-sized payoffs—so the distinguishing feature of a benign repeated game is not merely a continuation of play, that all potential cheaters are playing a *similar* game repeatedly.

A sustainable contract creates a game where honesty is always the dominant strategy in a game worth playing. Simplicity is crucial in generating good game theory equilibria because the state space grows exponentially in the number of players and actions they can take. An incentive-compatible contract avoids the more costly solution of establishing parties with ex-poste power to punish and confiscate.

[12] E.g., prior to commercial civil law there were courts along trade routes throughout Medieval Europe that enforced commercial laws (the *Lex mercatoria*), and its judgments were accepted not out of any legal authority granted by a state's monopoly on violence, but rather refusal would ruin one's business reputation and thus future revenue.

Oracle Data Submissions

Each betting period will contain up to 32 events and target a weekend (e.g., Friday night through Sunday night). Each contest is slotted into an array that can be unambiguously linked to its outcome via event logs that expose what events odds were on the contract. The schedule array contains a string with the sport (NFL, MMA, etc.), the two opponents, and the starting time. The favorite will be listed first and the underdog second.

The start time is necessary to prevent bets on started or completed games. Websites with event start times are tricky because sometimes these are listed as ET (i.e., New York City time); sometimes, they are automatically converted into one's regional time zone. It is best to buy an odds API for $50 or so, and these generally provide the start time and will reduce errors. Most data come with Greenwich Mean Time, GMT, often presented in ISO8601 date/time format, where the "Z" suffix means Zulu time, another word for GMT.

The following file format is sent via Python to the oracle contract, where the ellipses represent 30 other events for that epoch (the start times are in UTC, seconds since 1/1/1970).

```
{"_resultVector": [1,0,...],
"_teamsched": ["MMA:Buckley:Morono","NFL:Patriots:Broncos",...],
"_starts": [1697307477,1697311077,...]}
```

The **results** for the prior week are sent with the **schedule** and **start times** for the next week. The outcomes and schedules are bundled to minimize oracle effort, as these data are objective and unambiguous. Bundling the schedule and start times makes it easier for the oracle collective to anticipate and evaluate the subsequent odds data, as everyone can focus on these events. Bundling also reduces the number of data submissions to two, which creates some buffer in case a submission is rejected. If a settlement or initial post is rejected, then a settlement or initial post must be posted again, delaying the contract by a day, which is unfortunate but not tragic.

Another reason for combining the prior week's outcomes with the upcoming schedule is that these data are unambiguous. This makes the settlement/schedule data submission qualitatively similar in the evaluability. It also gives the oracle collective time to evaluate the less objective odds data, in that they can focus on the 32 events already identified.

Odds are sent on Thursday or Friday night, allowing betting to start on Friday or Saturday morning.

```
{"_decimalOdds": [605,230,...]}
```

The contract then repeats the process at the beginning of the following week.

Odds in the contract

The odds format is based on two primitives. First, the reciprocal of decimal odds is the fair probability of winning that would make the bet have zero expected value.

$$prob(win) = \frac{1}{decimalOdds}$$

The second is that the sum probability of betting on both teams equals "1 + vig', or 1.048.

$$prob(win;teamA) + prob(win;teamB) = 1 + vig$$

Thus, for an even bet, the odds for both teams would be 52.4%, as 0.524+0.524=1.048. Both teams cannot have 52.4% win probabilities, so the 2.4% premium reflects the vig and tells one how much of an edge one needs to beat the house.

We need the LPs and oracle to split the vig. This is done by setting the vig at 2.25% in terms of the odds the LPs experienced in the betting contract. This implies the even bet probability of winning is 51.2%. The contract applies this logic at the time of the bet with the following logic

```
    if (_team0or1 == 0) {
      betPayoff = ((1e7 / (512 + probSpreadDivBy2) - 1e4) * _betAmt) / 1e4;
    } else {
      betPayoff = ((1e7 / (512 - probSpreadDivBy2) - 1e4) * _betAmt) / 1e4;
    }
```

Here, the constants 1e7 and 1e4, as well as the number 512 instead of 0.512, accommodate the absence of floating point arithmetic in Solidity.

When a bettor wins, his payout is then taxed at 5%. As payouts on average are 50% of the total amount bet on both teams, one-half of 5% is 2.5%, which is the total vig in the contract, 4.8%. This is applied at redemption via the code:

```
      if (outcomeMap[epochMatch] != lose) {
        payout += betContracts[_subkId].betAmount;
        if (outcomeMap[epochMatch] == win) {
          payout += (betContracts[_subkId].payoff * 95) / 100;
        }
      }
```

And is applied at settlement to create the oracle dividend via

```
        uint256 oracleDiv = ORACLE_5PERC * uint256(winningsPot);
```

Here ORACLE_5PERC adjusts the winning payoffs by multiplying it by 0.05. It is 5e12 merely because we are adjusting decimals (i.e., 1 ETH is 1e4 in the contract but 1e18 on the blockchain. So, 5e12 × 1e4 = 5e16, 5% of 1 ETH.

By sending a single number representing the difference in win probability between the favorite and underdog, we get a number from zero to one. ASB adds this probability spread, divided by two, to the base 51.2% probability for the favorite and subtracts the same 'probability spread divided by two' from the win probability of the underdog. One then takes the reciprocal to get the decimal odds for the favorite and underdog.

The ASB repo contains an MS Access database that takes raw decimal odds data and transforms it into the probability difference divided by two. When pasted into its Excel spreadsheet, it generates the data in the correct format for sending via a Python program. The basic algorithm is this:

1. Take an initial set of odds from a survey of sportsbooks
    - Home team: +135 moneyline, 2.350 decimal
    - Away team: -150 moneyline, 1.667 decimal
2. rearrange so that the favorite team is first
    a. team[0]: 1.667, team[1]: 2.350
3. Translate into win probability
    a. Prob(win) = 1/decimalOdds
    b. team[0]: 60.00%, team[1]: 42.55%
4. calculate probability spread/2
    a. spread = 0.60 – 0.4255 = 0.1745
    b. spread/2 = 0.0872
5. Calculate new favorite, team[0], prob(win)
    a. prob(team[0] win) = 51.2% + 8.72% = 59.92%
    b. prob(team[1] win) = 51.2% - 8.72% = 42.48%
6. Translate prob(team[0] win) into decimal odds.
    a. Gross decimal Odds(team[0]) = 1 / 0.5993 = 1.669
    b. Gross decimal Odds(team[0]) = 1 / 0.4248 = 2.354
7. Adjust for the 5% oracle fee applied to winnings
    a. Net decimal Odds(team[0]) = 1+0.669*0.95=1.635
    b. Net decimal Odds(team[1]) = 1+1.354*0.95=2.287

In this example, the vig is 4.66%: 1 - 1.635 * 2.287 / (2.287 + 1.635). The above algorithm generates a vig near 4.5% across the range of odds covered in this contract. The tricky part of the above algorithm is that the raw odds spread, in the probability of a win, is added to 51.2%. If there were no oracle payment, this number would be 52.5%.

The website AVAXsportsbook.com displays the decimal odds users receive—net odds—if they win, though the odds are stored as the win probability spread divided by two times 1000.

LP Accounting

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:

$$\text{LPshares} = \text{AVAX invested} \times \text{TotalLpShares} / \text{TotalLpAvax}$$

For example, assume the contract has 123 AVAX owned by its LPs, who have 100 shares. This AVAX may be 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 Accounting

Oracle token holders must deposit their tokens in the oracle contract to vote and 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.

$$feePool_t = feePool_{t-1} + \frac{oracleRevenue_t}{TokensInOracle_t}$$

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

$$PotentialTokenRevenue_t = tokens_{t-1} \cdot \left(feePool_t - feePool_{t-1} \right)$$

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

$$OraclePayout_i = \frac{votesCast}{VotesPossible} \cdot PotentialTokenRevenue_i$$

$$OraclePloughback = PotentialTokenRevenue_i - OraclePayout_i$$

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 always equal to AVAX in the contract.

Margin Adjustment for New Bet

The contract tracks three types of Margin, all held in the array variable' Margin.'

LP Capital: AVAX owned by the LPs, free *and* locked up as collateral.

LP Locked Capital: This is AVAX owned by the LPs unavailable for bookie withdrawal. It represents the gross worst-case scenario loss for the LPs.

Bettor Capital: These are bettor funds applied to outstanding, taken bets. Bettors do not receive cross-margining.

New bets that increase the contract's net exposure will increment their LP locked-capital account, Margin [1]. Bets that decrease the LP's net position will decrease Margin [1].

This is calculated at bet time, and the LP's capital is moved into or out of Margin [2] depending on whether the bet increases or decreases the LP collective's net exposure. For example, an initial bet will increase the required Margin, but a subsequent small bet on the opposing team would lower the required Margin. A bet could move the book so that the net LP liability switches from team 1 to team 0 or consists of the decrease in the net liability on team 1. In any case, the above algorithm captures the difference in the worst-case scenarios for contract liability.

In this way, the LP's total book exposure is cross-margined so that 1.0 AVAX capital can support many bets via incremental bets on both contestants. At settlement, the locked LP capital, minus the accrued bettor payouts, is returned to the bookie's total capital, and the LP's locked capital account is set to zero.

For a team with decimal odds of 1.957. the total payoff for a win can be separated into two components: 1 + 0.957, the latter term representing the bettor's net profit and the former term representing the bettor's bet amount.

A bet is stored with two numbers: the bet amount and the bet's payout. For example, if one bet 123 AVAX on team A that has decimal odds of 3.00, the data would be:

Amount bet = 123

Payout = $2 \times 123 = 246$

Odds are stored such that.

$$\text{gross decimal odds on } team[0] = 1 + odds / 1000$$

LP Required Margin is the sum of the maximum liability for all the events in an epoch. Each event is independent, so the book is correctly margined by correctly margining all the individual bets. Thus, we need merely describe how margining occurs for a single event, knowing these are then summed for determining the overall Required Margin.

The total amount owed if team 0 wins equals the sum of the bet amount and its payoff for all the bets taken on team 0. Let us define two types of capital used to pay bettors: the payout or profit, which must come from someone other than the bettor, and the bettor's initial bet amount, which is returned with his profit:

$$winSum_0 = \sum_j betAmount_0^j \cdot (decOdds - 1)_0^j$$
$$betSum_0 = \sum_j betAmount_0^j$$

betSum$_0$ is the total amount bet on team 0, summing over all the bets on 0 (j loops through the bets). Bettor funds are available for payout but not part of the LP's Required Margin (in Margin [1]). **winSum$_0$** is the sum of the bettor's profit if team 0 wins, which requires AVAX from the LPs or bettors taking the other side. As the betSum of team 0's opponent, team 1, is available for winSum$_0$, the trick is monitoring the ability to cover the LP's liability given the amount bet on its opponent. This generates the following maximum liability for the LP (*aka* required capital) for a contest in that it is the maximum liability to either team in a contest:

$$\max\left\{winSum_0 - betSum_1, winSum_1 - betSum_0, 0\right\}$$

We add the zero term because the house will have only non-negative liability on every contest. For a new bet long on team 0 playing, the new bet and payout are added to the above max() equation and compared to the extant maximum liability. The difference is the change in the LP's required margin (margin[1]), which is offset by a change in the LP's free Margin (Margin [0]).

$$\Delta RequiredMargin = \max\left\{winSum_0 - betSum_1 + newPayout_0, winSum_1 - betSum_0 - newBetAmount_0, 0\right\}$$
$$-\max\left\{winSum_0 - betSum_1, winSum_1 - betSum_0, 0\right\}$$

The maximum bet size is displayed within the GUI when a user toggles the radio button. It is calculated using the following logic. We use the superscript *i* for the pick and -*i* for the opponent. The potential liability for pick *i* is

$$liab^i = payOut^i - bets^{-i}$$

The global maximum exposure for any match is a function of the amount of LP capital and the concentration factor. This number applies to each new bet, capping the LP's exposure to any single event.

$$maxExposure = lpTotalCapital / concentrationFactor$$

The amount of available LP capital is

$$freeLpCapital = totalLpCapital - lockedLpCapital$$

With these data, and the odds offered on the bettor's pick, we can calculate the change in LP locked capital on a new bet:

Maximum exposure for a pick:

$$maxExposureAvailable^{i} = \max\left\{liab^{-i} - liab^{i}, 0\right\} + \max\left\{lpFreeCapital, maxExposure - max\left\{liab^{i}, liab^{-i}\right\}\right\}$$

To translate this into a bet size, we divide by the payoff odds. For example, if the odds were 1.500, this pays out 50% on each dollar bet. Thus, with 1.0 in LP exposure available for the pick, that would allow a bet for 1/0.5 or 2.0.

$$maxBet^{i} = \frac{maxEposure^{i}}{\left(decOdds - 1\right)^{i}}$$

LP exposure across matches is independent. The assumption for LP exposure is the worst-case scenario, so there will be no chance of insolvency, as a bet cannot be taken without capital available.

Cheat scenarios

Arbitrage ensures odds do not vary by more than 3% when translated into a probability of a win, which intuitively might lead to bad odds posted under statistical plausibility. Such a subtle odds cheat would not work for the same reason counting cards in blackjack only makes sense if you can play several hundred hands. Biasing the odds to get a 1% edge is defensible as a measurement error (by the poster or validator), but only on a couple of events, which would generate an unattractive Sharpe ratio (the volatility would be much higher than the expected return). Biasing odds on dozens of games week after week would stand out like posting 3:1 odds on a single even-money game, and rational liquidity providers would exit before the cheating oracle could make a thousand bets needed to monetize a 1% cheat.

The more obvious way to cheat would be to report an incorrect winner. This involves the oracle, but who would they cheat, LPs or bettors? If they try cheating the bettors, they will have to create an LP sock-puppet account and then set the outcomes to favor the LP's positions. The cheat benefit would be shared with the other LPs, diluting the cheater's take, and as the LP exposure is passive, they could not maximize it. In contrast, if the cheating oracle created a bettor sock-puppet account, they could maximize their take by maxing out LP exposure and not have to dilute their theft with unaffiliated bettors.

If outsiders see an incorrect outcome processed, it could be a simple mistake, as perhaps the event was unpopular and there were not many bets on it, so the oracle didn't bother with double-checking the data. A real cheat would also show unusual betting on the same side as the cheat. In such a case, it would be rational for bettors never to use the dapp again. For retail bettors, they should not have too much money at stake, and whether they were on the wrong side of this cheat would be random. For the LPs, the outcome would be more severe, and this is why it is helpful to have an LP-oracle overlap, as this would give the oracle greater incentive to be honest.

Settlement Detail

Settlement records which bets won and then allocates bettor and LP capital to accounts that ensure accrued accounts are fully collateralized. Each bet creates a struct that contains the team and week of the bet. These two inputs create a hash mapped to a number representing its game outcome: 0 for a loss, 1 for a tie, and 2 for a win. When the array of 32 results is sent to the settlement method, the mapping is created (the mapping is zero for uninitialized hashes, so unless updated, the mapping is 0). This mapping is then used for redemptions, in that a bettor claiming his winnings will need the {epoch, match, team} hash to map to a 1 or 2 to generate a payout.

In addition to creating non-zero hash mappings for non-losing teams, the total payments to all bettors were generated using the results and the paySum and betSum arrays:

$$WeeklyPayBack = \sum_{i=0}^{31} 1_{WinOrTie}(i) \cdot betSum_i$$

$$WeeklyWinnings = \sum_{i=0}^{31} 1_{Win}(i) \cdot winSum_i$$

Here 1_x is an indicator function that is 1 if true, 0 else. The *WeeklyWinnings* represents the bettor profit, while the WeeklyPayBack represents the initial bettor funding. The oracle fee of 5% is applied to the bettor winnings, representing about 2.5% of the total bet amount. As individual payouts are less than or equal to the total payouts in any week, rounding truncations on individual redemptions will not compromise contract solvency; rounding will not prevent redemptions.

At settlement, accounts are adjusted as follows:

Redemption capital = WeeklyPayBack

PayoffPot = WeeklyWinnings * 95 /100

Oracle fee revenue = WeeklyWinnings * 5 /100

The bookie's capital then adds the money bet that week minus the payouts for wins and ties.

bookiePool = bookiePool + bettorLocked – redemptionPot – payoffPot

The oracle revenues are then just 5% of the WeeklyWinnings and are transferred to the oracle contract in the settlement function.

The bettor's money exists in the residual and must be claimed via redemption. At redemption, their winning bets are credited to the bettor's user balance, available for withdrawal or future bets.

After settlement, the LP's locked Margin is set to zero, so all LP funds are available for withdrawal.

Gas for transactions

contract	function	gas (x1000)
oracle	settleRefreshPost	350
oracle	settleRefresh voteProcess	909
oracle	odds Post	112
oracle	update voteProcess	75
oracle	vote	37
oracle	deposit Tokens	52
oracle	wd Tokens	84
bet	bet	126
bet	deposit as LP	54
bet	deposit as bettor	48
bet	wd as LP	44
bet	wd as bettor	33
bet	LP claims token Rewards	62
bet	bettor redeems 1 bet	40
bet	bettor redeems 16 bets	161

Simple Restrictions in the Contract

The contract prioritizes simplicity to minimize attack surfaces and make the oracle's job as easy as possible. These are complementary objectives, as a simple contract implies fewer options, decisions, and the size and scope of data to validate. Initially, the oracle administrators will be working on faith, which is tolerable as long as their duties are not too difficult.

Odds Restrictions

- **One odds number in the contract**

Standard odds are presented as a pair, with a spread so that simultaneous bets on both teams lose money for the bettor and make money for the house. A prominent attack surface for a smart contract would be for the odds to imply an arbitrage, as the offsetting bets would generate a sure profit, enabling the hacker to drain virtually all the LP's 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.

- **No extreme odds**

Matches with extreme underdogs (e.g., 10-1) are attractive for hackers, as they generate the most revenue for the smallest amount of capital. They also require too much LP capital. Initial decimal odds on favorites greater than 5.65 are not accepted. Such matches will not be covered. This would eliminate about 5% of NFL games historically but is common among college football and MMA. This also eliminates bets like "who will win the next golfing event."

Oracle Submission Restrictions

- **Weekly betting cycles**

Standard centralized sportsbooks cover diverse events on most days of the week, including exotic bets that are not straightforward to validate. This demands a significant 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.

Games are constrained to start between the following Friday and Tuesday; settlement cannot occur until Monday. Thus, it is impossible to generate two settlements within a week. This reduces the vigilance needed to assess the oracle.

- **Maximum of one daily submission, with 12 hours to evaluate.**

The oracle processes at most one submission per day, which must be submitted during the first three hours of GMT. This makes it easier for the oracle to keep track of the data it must evaluate. Odds movement within a week is generally within the effective bid-ask spread implied by the standard

sportsbook vig applied in ASB, so a Thursday or Friday odds submission should provide sufficient protection against the adverse selection risk created by stale odds (outliers can be halted by the oracle).

As no healthy adult sleeps more than 10 hours a day, the 12-hour window between 3:00 and 15:00 GMT ensures token holders can evaluate and vote before the data submission is processed, regardless of their time zone. The objective is to make it feasible for a single person to do this manually without an extreme investment. The primary functions can be automated to a great degree, and Python programs for processing and submitting oracle data are provided in the GitHub repo. The crucial issue is time, giving the oracle collective time to evaluate the data carefully.

- **Submissions are not allowed on Saturday and Sunday.**

The oracle cannot settle for at least 48 hours after Friday evening, which makes the oracle's duties less onerous. The oracle can halt betting on a match if the odds change significantly before an event. This moves the start time back four days, and as betting is only possible before the start time, it precludes betting. Maximum 32 events

The settlement function loops through the events, and 32 is big enough to capture the weekend's biggest events. 32 is also not so big as to not create a burdensome validation process, as obscure contests would be more challenging to assess.

- **Easily anticipated games**

Football, boxing, and MMA will be the primary focus, as matches and odds are well-publicized early in the week, making it easier for the oracle collective to assess odds data. High-profile events other than football, boxing, and MMA can be accommodated on a case-by-case basis (for example, a World Cup soccer match).

Simplifications

- **Only three contracts**

This dapp consists of three solidity contracts: betting, oracle, and token. In contrast, Uniswap's V3 'contract' contains 31 contracts, which makes it virtually impossible for most people to audit. With ASB, one can download the three contracts and test different scenarios to find a hacking surface. I provide a dozen hardhat tests as templates to build upon in my GitHub repo.

- **Contracts not upgradeable**

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

- **No adjudication process**

The oracle incentives are based on the present value of the oracle token, which should be sufficient. Redundant mechanisms do not add to this incentive, as the adjudicators would require payment, which reduces oracle revenue and thus the value of oracle tokens, reducing the oracle's incentive.

- **Everything in AVAX, no stablecoin**

By using native AVAX for all bets, we eliminate costly swapping into and out of stablecoins. As stablecoins are generally centralized, we eliminate an attack surface as well, as one could imagine Circle, at the behest of some US regulator, preventing the betting contract from receiving or sending USDC.

Users will have to bear AVAX price risk, but this is a minor inconvenience relative to the extra costs created by a stablecoin.

Oracle Restrictions

- **No token withdrawals during a vote**

prevent accounts from double voting.

- **Oracle has minimum and maximum token requirement.**

The minimum requirement makes the cost of evaluating or sending data less than the expected loss from a reputation-destroying fraudulent data submission. The acceptable range is from 10 to 22% of the lifetime supply (1 million).

- **Oracle token holders are only paid if they vote**

This motivates the oracle token holders to evaluate the data, though indirectly. Oracle token depositors earn oracle revenue based on their proportion of the tokens in the contract when the oracle receives revenue. That is then adjusted based on their voting proportionate way (e.g., if the token depositor votes 75% of the time, they will receive 75% of the income they earned via their proportionate token deposits).

- **An oracle account cannot send consecutive data consecutively.**

If one token depositor sends the settlement/new schedule post on Tuesday, he cannot send the odds data later that week; if he sends the odds data, he cannot send the subsequent settlement post. This motivates all oracle members to create their own data submissions, as they cannot depend on a single oracle member who might come to dominate data submissions. While this could devolve into two oracle accounts alternately posting data, it is a slight 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.

- **Token holders cannot send data for three epochs after a rejected submission**

If an evil token holder tries to hack the contract, it is in the best interest of the other token holders to reject the data submission. If the evil token holder were a computer wiz and was able to post data before anyone else, and he became frustrated, he might want to effectively DDOS the contract by always sending bad data, which is then always rejected. With this limitation, the evil token holder is put into a 'time out' situation, allowing the remaining token holders to process that week's bets. The evil token holder could submit insufficient data every four epochs, but it would be pointless.

- **Token holders can only adjust the concentration limit or halt an event's betting activity once an epoch**

This is to mitigate the damage by trolls, who might find wrecking the contract amusing. The token holder cannot withdraw their tokens for an epoch to prevent an obvious mechanism to subvert this restriction.

LP Restrictions

- **LPs are charged a 1% fee if they withdraw in the same epoch**

Suppose LPs could deposit and withdraw quickly without a fee. In that case, some LPs might find it profitable to scare away other LPs by depositing large amounts 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 would not help the LP collective, so they are explicitly discouraged via this fee.

- **LPs cannot withdraw in the same epoch they claim token rewards**

This prevents LPs from claiming rewards multiple times. There is a fixed amount of token rewards in the betting contract.

- **LPs cannot fund or withdraw when betting is active**

This prevents LPs from avoiding or taking advantage of information about bet outcomes as they accrue.

- **LP capital exposure is limited on any one match.**

If the LPs have 100 ETH, an adjustable concentration factor prevents the LPs from having an exposure greater than 100/concFactor in any event.[13] This helps control diversification risk. The concentration factor is constrained to be between 2 and 16, as the optimal number will be determined by experience.

Betting Restrictions

- **Min bet size 1 avax**

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

- **No betting after the game start time**

Obviously, a match with fixed odds is easier to bet on when it is partially completed. The ambiguity of block time 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 the oracle chooses. For example, if x=10 and total capital were 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 no more free LP capital would be left.

Emergency mechanisms

There is no outside adjudicator to rectify problems, as this would delay payments and complicate the contract—how to incent the adjudicator? All problems must be solved on-chain within these contracts.

A match's odds may become stale. Allowing the oracle to turn such matches off does not expose the contract to mischief; it just prevents more bad trades from happening (trades already taken would still be valid).

If an initial data submission is rejected, the week is not ruined. A replacement can be made the following day, allowing the contract to function that weekend. There is no penalty for rejected submission, as the oracle has the time to make a fully informed decision, and it is irrational for the oracle to choose to cheat

[13] This number is arbitrary, and experience will reveal the best parameter. This protects LPs as diversification lowers risk in the standard way--This factor can be adjusted, as experience will generate useful information on the best diversification factor.

deliberately. However, unintentional errors are natural, often not seen until one sends the data to the contract.

Off-chain odds can change quickly and significantly, exposing the LPs to bets with an objectively negative expected value position. In that case, oracle token holders can immediately pause new betting on an event. This action does not require the usual 12-hour vetting period to allow oracle token voting. It does not expose LPs or bettors to more risk, just preventing new bets on those matches. Such an action has no upside if this is not true for the oracle.

If bad odds or starts are submitted to the betting contract, the oracle collective could nullify this action by posting a result of a tie regardless of the outcome. This allows the LPs and bettors to get their money back as if nothing happened, and the 'incorrect' but fair tie result should be explainable by the event logs showing the earlier hack. This would be an extreme scenario, like a fork in a blockchain, but it is always good to anticipate a worst-case scenario.

Oracle submitters can resubmit in the three-hour window they have to submit data, and other oracle depositors cannot vote until the window has expired. This is aimed at the case where a submission contains an obvious error that the submitter did not notice until it was posted, something I experienced while running a beta version of this contract. This still gives the oracle collective 12 hours to evaluate the data.

Decimals

The ASB oracle token has 3 decimals. AVAX has 18 decimals, just like ETH. Within the betting contract, however, AVAX has 4 decimals. Thus, if you deposit 1.23405654 avax it is recorded as 12340. The frontend adjusts for this so that a user typing 1.234 into the bet is betting 1.234 AVAX, but if using Remix or Python, one must send 12340. LP shares have no decimals; they are not tokens and are not transferable.

Contract Data

Contracts are verified and active on the Avalanche C-chain. All contracts compiled in v0.8.19, optimization enabled with 2000 runs. They are in the GitHub repo, and one can access them via Remix or Python.

https://avascan.info/blockchain/all/address/0xB73Cb2696726b7356e03c697672e2Dcc751407D0/contract

https://avascan.info/blockchain/c/address/0xD8Fc0B73066D090520428e4F6809be92af9fda95/contract

https://avascan.info/blockchain/all/address/0x43B8B88f5f0193B2dc86723D6BC515ACF424F917/contract

download all into Remix via the gist

https://gist.github.com/efalken/1f658d097963f0d8e690e871685d7fec