TezBet: A Decentralized Approach to Sports Betting

Eloi BESNARD, Enguerrand DECLERCQ, Jean-François DUMOLLARD, Victor LAFORET

19 december 2021

Abstract

TezBet is a decentralized application built on the Tezos blockchain. It offers a self-sufficient alternative to common centralized soccer betting services. Using the zero-sum game principle, it does not need to store any backing liquidity to pay the winners, for their share is calculated upon the total bet amount. Using this mechanism, odds manipulation and bankruptcy are both made impossible.

1 Introduction

Most sport betting services rely on centralized third parties for both odds computing and gains distribution. On one hand, the players' spendings flow through a company's account before being redistributed accordingly. This means that such companies could arbitrarily choose to keep the players' spendings for themselves. On the other hand, the way odds are calculated remains vague. While they are supposed to reflect the mere likelihood of an outcome, they are mostly meant to keep betting companies sustainable. Ultimately, data collection through KYCs is inevitable using this kind of services.

2 Game rules

The TezBet smart contract enforces specific rules. These rules are immutable once the contract is deployed:

- Only non-started games accept bet creation and removal;
- Players are allowed to bet on several games simultaneously;
- Bet removal comes with a service fee;
- Bets on ongoing games cannot be removed or modified;
- Ongoing games cannot accept new bets;
- Players cannot redeem their gains before the final score of the game has been set.

3 Odds

3.1 Theory behind odds: zero-sum game

A zero-sum game [3] is a mathematical representation in game theory and economic theory of a situation in which an advantage that is won by one of two sides is lost by the other. If the total gains of the participants are added up, and the total losses are subtracted, they will sum to zero. Thus, cutting a cake, where taking a more significant piece reduces the amount of cake available for others as much as it increases the amount available for that taker, is a zero-sum game if all participants value each unit of cake equally. In the markets and financial instruments, futures contracts and options are zero-sum games as well.
In contrast, non-zero-sum describes a situation in which the interacting parties' aggregate gains and losses can be less than or more than zero. In our example, most betting services are non-zero-sum games: they take profit from the players' losses and cover the winners' gains through their treasury.

The way we leverage zero-sum mechanism is as follows:

Let n the number of players.
Given $a_{k,j}$ the amount bet on outcome k by player j, the multiplier of the bet amount for this outcome k, can be stated as

$$G_k = \frac{\sum_{j=0}^{n} (a_{TeamA,j} + a_{TeamB,j} + a_{Tie,j})}{\sum_{j=0}^{n} a_{k,j}} \tag{1}$$

Therefore a winning player j who bet $a_{k,j}$ XTZ on outcome k will receive $G_k \times a_{k,j}$ XTZ.

3.2 Simulating potential gains

Table 1 is an example of potential gains for a fictive game in which Team B wins. This scenario shows the total bet amount by each player and their returns by the end of the game according to their choice.

Table 1: Simulation of a game in which Team B wins

Player	Bet amount (XTZ)	Choice	G_k	Return (XTZ)
1	7,500	Team B	2.625	19,687
2	2,000	Tie	10.5	0
3	10,000	Team A	1.9	0
4	1,000	Team A	1.9	0
5	500	Team B	2.625	1,312

4 Leaderboard

The leaderboard displays the all-time 10 highest earnings. Only finished games values are shown. A same player could appear several times in the leaderboard.

The leaderboard is stored in the main smart contract. Given that game records are deleted from the contract storage once winners have redeemed their XTZ, a specific field storing the 10 highest earnings is needed.
Everytime players redeem their gains, the contract checks whether the bet qualifies to be added to the leaderboard.

5 Implementation

5.1 Implementation constraints

- Games being eventually deleted from contract storage, no history is kept after their deletion;
- Games status are automatically set through timestamps and cannot be set manually [2].

5.2 Lifecycle of a game

5.2.1 Game status

Each game is initialized in the contract storage with a unique ID, a non-started status and an empty bets record. During this phase, players can make and remove bets. Once a game has begun, the total bet amount is locked within the game's pool. When it ends, winners can redeem their gains while losers cannot withdraw the amount they bet. Once winners have redeemed their gains, the game record is deleted from the contract storage.

5.2.2 Rewarding early betters

Odds being fixed when a game starts, players will be tempted to wait and see the most likely odds before placing their bets. Doing so, players would be acting as bidders speculating over the final odds. However, this behaviour promotes indecision and needs to be addressed. Otherwise, they would be no incentive to be the first better and players would end up not betting at all. Furthermore, betting on outcome A when there is no bet on competing outcomes B and C is prohibitive:

- Outcome A betters lose their bet if outcome B or C occurs;
- Outcome A betters simply recover their bet amount if outcome A occurs.

In these scenarios, it is better not to bet at all. Therefore, built-in mechanisms rewarding early betters are needed. They come as follows:

- Outcome A betters recover their bet if outcome B (resp. C) occurs and nobody bet on outcome B (resp. C);
- The bet removal service fee depends on the bet timestamp: the earlier the bet, the lower the fee. This gives early betters an advantage over late betters.

The service fee y_s equation is given by 2, with $x = t_{gamestart} - t_{bet}$. Given this equation, $y_s(0) = 0.2$ and $y_s(-86400) = 0$: when a player bets 24 hours ($t_{bet} = -86400s$) before a game begins ($t_{gamestart} = 0s$), his bet removal fee equals 0XTZ. However, a player betting 12 hours ($t_{bet} = -43200s$) before a game begins will pay a 10% fee for each bet removal.

$$y_s(x) = 0.0000023148148x + 0.2 \tag{2}$$

5.3 Game IDs generation

Contract-stored game IDs are identical to the API game IDs. Having a single ID for both off-chain and on-chain games binds them together and eases their usage on both front and back ends.

6 Hybrid data collection

6.1 Oracles

Blockchain oracles are entities that connect blockchains to external systems, thereby enabling smart contracts to execute based upon inputs and outputs from the real world [1].

Oracles provide a way for the decentralized Web 3.0 ecosystem to access existing data sources, legacy systems, and advanced computations. Decentralized oracle networks (DONs) enable the creation of hybrid smart contracts, where on-chain code and off-chain infrastructure are combined to support advanced decentralized applications (dApps) that react to real-world events and interoperate with traditional systems.

Each oracle acts the same way a standard API does; the main difference is that data is fetched and currated by a network of independent validators. Decentralized oracles network Chainlink aggregates data from various nodes scraping the same API. Ultimately, Chainlink uses different sources aka oracles for each data collection, which means that a single spoofing oracle will not affect data integrity. Using different oracles also helps us taking advantage of single point of failures.

6.2 Contract storage

To be truely self-sufficient, TezBet cannot rely on a single API because of potential data spoofing. Therefore, TezBet uses a decentralized network of sport data oracles provided by Chainlink. These oracles allow TezBet to define whether the games are playing or not and fetch the final score, while preventing potential data spoofing from a single oracle.

Besides, the games timetables are stored in the contract storage in a timestamp format. It allows TezBet contract to prevent users from betting after the game has begun without making additional oracle calls.

6.3 Frontend display tradeoff

Looking for the next games and their live scores would constitute costly and gas-intensive oracle calls. Besides, we would need a bot to consume oracles and update the contract storage. This would be possible using Chainlink Keepers - for instance - if it existed on the Tezos main and test nets. Instead, we are using standard APIs to look for the next games and their live scores. This tradeoff allows us to minimize transaction fees while displaying the data players need on the frontend. Although, the games status and their final scores are checked through oracle calls.

References

[1] Chainlink. What is a blockchain oracle?

[2] SmartPy Docs. Reference manual - smartpy.io.

[3] Wikipedia. Zero-sum game.