Fortuna: A Novel Staked Voting System for Distributed Pari-Mutuel Gaming

Tucker Moore
Georgetown University
Washington DC, United States
tspaightm@gmail.com

Nathan Marshall
Georgetown University
Washington DC, United States
nathan.douglas.marshall@gmail.com

Dr. Eric Burger
Georgetown University
Washington DC, United States
eburger@standardstrack.com

Abstract—**In 2008, Bitcoin was introduced to the public and became the first widely accepted, fully decentralized cryptocurrency. However, as effective as blockchain technology has proven to be, one challenge that still exists is the ability to authenticate real-world event outcomes. Previous approaches rely on "trusted" feeds, self-reporting, and semi-centralized oracles – all of which have deficiencies. What is needed is a system and method of bringing real-world data into a decentralized network in a robust and low-cost way. One application for such a system is gaming. Casinos offer many benefits to a bettor. However, players often face large fees and unfavorable odds in order to participate in this centralized process. These high fees and unfavorable odds provide both an opportunity and a need for a decentralized platform with the ability to provide players with a more favorable means to engage in gambling. We seek to solve this problem by implementing a novel method of staked voting, enabling a network of otherwise trustless users to reliably agree on the outcome of a real-world event, such as the 2022 World Cup. We require winning bettors to vote on the outcome of another event in order to redeem their winnings. If they vote with the consensus, their winnings will be released. Otherwise, their winnings will be forfeited. This method of staked voting creates a fully distributed, self-regulated, pari-mutuel betting platform.**

Keywords—blockchain, consensus protocol, cryptocurrency, decentralized applications, smart contracts, sports

I. Introduction

A. General Background

In 2008, Bitcoin was introduced to the public and became the first widely accepted, fully decentralized cryptocurrency. It was the first cryptocurrency to combine blockchain technology with the concept of proof of work [1] to address the problem of double spending [2]. Since then, thousands of other cryptocurrencies have been created with almost one thousand such currencies trading daily [3].

However, as impactful as blockchain technology has been, a challenge that still exists is the ability to bring real-world data onto the blockchain in a reliable and trusted way. Users can only access data reliably created on the blockchain [4] and the "truth" is limited to what the ledger can verify with no mechanism to verify data from the real world. There is a clear need for such a mechanism without a centralized point of failure and with the same security and veracity expected of a blockchain.

We hypothesize that, by implementing a novel staked voting algorithm and a unique pari-mutuel design, we can create a distributed protocol that satisfies the following: (1) no centralized point of failure, (2) events with any number of potential outcomes can be determined with the same level of trust, and (3) resistance to expected attacks such as Sybil attacks, collusion, denial of service, and hash manipulation.

B. Motivation

Although there is a need to bring real-world data into a blockchain in a reliable and robust way, most decentralized blockchains have no mechanism wherein users can validate the outcome of real-world events. To address this problem, blockchains often introduce a special class of users known as oracles [5].

There are many different types of oracles, but all serve the same purpose – to transfer data between the real world and the closed-world system of the blockchain [6, 7]. A significant challenge in incorporating oracles on a blockchain, especially a decentralized blockchain, is maintaining trust across all users. There must be an incentive in place for each oracle to act honestly. While there are several oracle platforms [8, 9, 10], the current state of the art has significant deficiencies: (1) there are major limitations on supported event types, (2) reporting algorithms are susceptible to attacks, and (3) the existing platforms exhibit varying degrees of centralization.

One application for such a network is gaming. Casinos offer many benefits to gamblers. They allow players to bet on games and events, determine event outcomes, and provide capital to manage payouts. However, players also face large fees, often masked as unfavorable odds, in order to participate in this centralized gaming process. Nevertheless, there is still an estimated $150 billion wagered on sporting events each year [11]. Due to the size of this market and the need for a decentralized platform with the ability to provide players with more favorable means to engage in gambling, gaming offered an intriguing use case for a distributed oracle network.

C. Contribution

Our contribution is Fortuna: a betting protocol implementing a new distributed oracle that leverages a novel staked voting algorithm and a unique pari-mutuel design. Fortuna offers a decentralized platform on which real-world event outcomes can be determined and provides users with a more affordable and favorable means to engage in gambling without the need for, or the costs of, a centralized authority.

II. Background

A. Cryptocurrency Background

A digital currency is a currency without a physical form and is only transferrable electronically [12]. Traditional digital currencies are centrally issued and regulated by banks or, in some cases, by privately owned companies. A cryptocurrency, however, is a type of digital currency that is typically decentralized, requiring no central regulating entity [2]. Decentralized cryptocurrencies use cryptography and a public ledger system to ensure honesty among users. Decentralization allows distributed security across its users and faster settlement of payment [13].

In 2008, Bitcoin was introduced to the public and became the first widely accepted, fully decentralized cryptocurrency. It was the first cryptocurrency to combine blockchain technology and proof-of-work in a way that provided a fully decentralized network the ability to timestamp transactions and prevent double spending [2]. "Double spending" occurs when a user places two transactions with two validating entities at the same time, referencing the same coin signature. The blockchain's ability to timestamp transactions by successively chaining blocks of transactions together prevents two conflicting transactions from being published – "chaining" requires one transaction to be processed before the other, thereby invalidating the second transaction [14].

Since the inception of Bitcoin, thousands of other digital coins and tokens have been created with a total market capitalization of approximately $340 billion, with almost one thousand trading daily [3]. As impactful as blockchain technology has become since Bitcoin's release in 2008 [12], a challenge that still exists is the ability to bring real-world data onto the blockchain in a reliable and trusted way. Historically, oracles have been used to solve this problem.

B. Blockchain Oracles

An oracle is a privileged entity with the power to transfer data between the real world and the blockchain. Although there are many types of oracles, most fall within three broad categories: software, hardware, and human [7]. One difficulty in incorporating oracles on a decentralized network is in maintaining trust across its users. There must be an incentive put in place for each oracle to act honestly. This is necessary to guarantee (with high probability) the consensus reached by a quorum of voters, or oracles, matches the "truth." Without an incentive to behave honestly, there is no reason to believe a real-world event outcome can be correctly incorporated into a blockchain.

The following three oracle constructions illustrate distinct architectures, though they follow the same general process: (1) an event is proposed to the network, (2) bets are placed, (3) the event occurs in the real world, (4) oracles report the event outcome, and (5) rewards/payouts are distributed.

1) ZenSports: ZenSports is a partially decentralized sports betting marketplace built as an application on the ICON blockchain [9]. While ZenSports provides a protocol to allow peer-to-peer betting, a large part of the protocol is operated by the ZenSports team. First, while the betting contracts themselves are created by users, they are limited to the events and bet types ZenSports chooses to support [15]. Second, although there are plans to decentralize the process in the future, at this time, dispute resolution is handled unilaterally by ZenSports [16]. This centralization within the platform precludes users from proposing less popular sporting events or non-sporting events altogether.

2) Wagerr: Wagerr is a semi-centralized betting platform made up of two user types: "standard wallets" and Oracle Masternodes. Standard wallets interact with the network, validate transactions, and form blocks of transactions to build the blockchain and earn rewards. Oracle Masternodes are responsible for publishing events to the network as well as voting on event outcomes. Although Wagerr implements a more decentralized design, it still limits the events on which users can place bets to certain binary outcome sporting events selected by the Oracle Masternodes [10]. Additionally, Wagerr relies heavily on a relatively small set of oracles to both propose events as well as report outcomes of those events.

3) Augur: Augur is a decentralized platform for prediction markets that awards users for correctly predicting future events [8]. Augur allows users to create markets representing specific events, and to purchase and sell "shares" of that market, representing the potential outcome of those events, acting as wagers between two users. Augur requires the user who created the market to designate a single "resolution source" for reporters to reference to determine a market's outcome. This introduces a single source of failure to the system. If each reporter uses the same designated source as truth, the system can become centralized and easily susceptible to attacks if a user were to gain control of that source. Additionally, because Augur allows oracles full control over their stake and the market on which they vote, it is susceptible to an attacker targeting a specific market with false votes.

C. Shortcomings

As detailed above, there are significant deficiencies and shortcomings with existing oracles. All oracles that we investigated only support events with two potential outcomes, while some place limits on supported event types and also exhibit varying degrees of centralization. The challenge is how to resolve each of these deficiencies – building a platform that (1) puts no limitations on user-proposed events, including the number of potential outcomes, (2) eliminates a separate, dedicated class of oracles or voters, allowing users themselves to report event outcomes in a trusted manner, (3) provides a distributed betting protocol that puts no limitation on bet type and size, and (4) removes all power from any centralized entity or entities.

D. Gambling Background

Casinos offer many benefits to a gambler. They create events on which to bet, set lines and odds for those events, determine winning outcomes, and manage payouts to winning bettors. However, there are inherent deficiencies in centralized casino operations. Both physical and online casinos have bloated infrastructures with predominantly manual operations and processes. Brick and mortar casinos have large overhead costs, including labor costs for dealers, bookmakers, and supplemental staff, that necessitate charging high fees to the players. These fees, often masked as unfavorable odds, ensure that in the long run, the player will lose, and the casino will win (i.e., "the house always wins").

III. Fortuna Overview

A. *Overview*

Having identified the specific deficiencies of existing oracles, Fortuna seeks to resolve those deficiencies and shortcomings by establishing a fully distributed pari-mutuel betting platform. Any user will have the ability to propose events, place bets, accept payouts, and – using a novel method of staked voting – help determine event outcomes, all without a centralized regulating entity. Moreover, Fortuna's pari-mutuel betting system [17] will remove all restrictions on the number of potential outcomes for an event.

In contrast with the traditional "transaction" often used in other cryptocurrencies, Fortuna utilizes a more general "action" that contains one or more "elements." Each element can represent a pool, a bet, a vote, or a transaction. A pool represents any event on which a user can place a bet. A bet is a wager that a user places on a potential outcome of a pool. A vote is cast to help determine a pool's actual outcome. And finally, a transaction is a transfer of coins from one user to another.

In a typical transaction-based cryptocurrency, each transaction requires a source address and a recipient address to ensure the network can verify both where the coin or token originated as well as where it was sent. In this way, a coin or token always has an owner. With Fortuna, by contrast, creating pools, placing bets, and casting votes requires only a source address in order to verify that the user can perform the action. Because Fortuna pools represent real-world events, there is a necessary time period between bet placement and outcome determination. During this time period, the coins wagered on the pool are in a state of holding where there is no coin owner. Once the pool has settled (i.e., the outcome has been established), the winning bettors are initially paid out in votes. To redeem these winnings, the bettor must place these votes, with the consensus, on the outcome of another pool. By this methodology, Fortuna defines a coin (as well as a vote) as the chain of signatures between action elements, similar to how a typical transaction-based cryptocurrency defines a coin or token as the chain of signatures between transactions.

B. *Protocol*

1) Creating a Pool: As Fortuna is a platform allowing users to place bets on real-world events, the first step in the process is to create a pool, representing a real-world event. Any user can create a pool and publish it to the Fortuna blockchain. However, to dissuade nefarious users from spamming the network with bogus pools, the pool creator, or "architect," must first contribute a small fee to the pool's pot. This fee is dynamic and based on the distribution of pool sizes. Additionally, to cover the initial fee and to encourage users to create legitimate and appealing pools, the architect receives one percent of all bets placed on the pool once it has settled. To create a pool (Fig. 1), the architect specifies a title, a description, a list of potential outcomes, and an expiration time for accepting bets. The title and description are used to provide information to the bettors as to what event the pool represents. The list of potential outcomes includes the outcomes for the event on which users can place bets. The expiration time sets the deadline, at which point the pool stops accepting bets and starts accepting votes. Finally, each pool is assigned a unique pool ID that is used to place bets and cast votes on that pool.

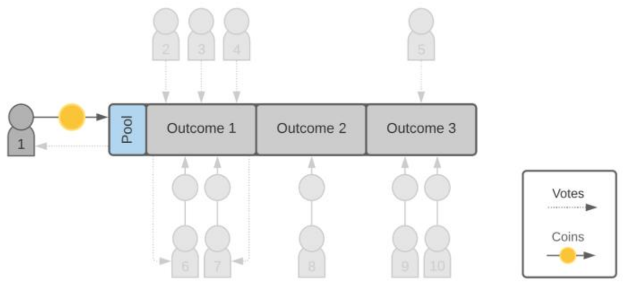

Fig. 1. This figure shows a user creating a pool with three potential outcomes. Before publishing the pool to the blockchain, the architect is required to contribute a small fee to the pool's pot.

2) Placing a Bet: Once a pool is published to the blockchain, it begins accepting bets. Any user can place a bet on any pool. To place a bet (Fig. 2), a user specifies the pool on which they wish to place their bet via its unique pool ID, the outcome on which the bet is placed, and the amount of the bet.

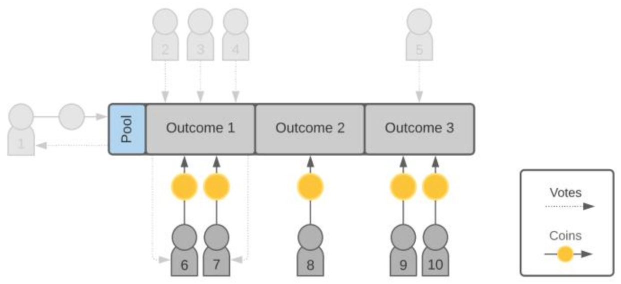

Fig. 2. This figure shows five users betting on a pool. Users 6 and 7 placed bets on Outcome 1, User 8 placed a bet on Outcome 2, and Users 9 and 10 placed bets on Outcome 3.

3) Casting a Vote: Once the expiration time for a pool is reached, the pool stops accepting bets and starts accepting votes. To cast votes on a pool (Fig. 3), a user specifies the pool on which they wish to cast their vote via its unique pool ID, the outcome of the event (i.e., their vote), and the number of votes to be placed. The pool will continue to accept votes until a quorum is reached. Once a quorum is reached (as defined below), the pool is settled and the outcome with the majority vote is accepted as the winning outcome by the network. Users who placed bets on the winning outcome are paid out in votes and are now required to place those votes with the consensus on another pool in order to redeem their winnings. Those who vote against the consensus on another pool will forfeit their winnings. Additionally, the pools on which a winning bettor may cast votes are randomized using voting partitions. This procedure maintains voter honesty and network integrity.

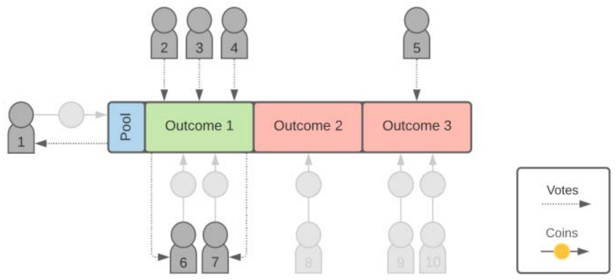

Fig. 3. This figure shows users voting on a pool's outcomes and other users being paid out in votes. Users 2, 3, and 4 vote on Outcome 1, and User 5 votes on Outcome 3. Because Outcome 1 collected the majority vote, it is deemed the winning outcome. Users 6 and 7, who bet on Outcome 1, are paid out in votes. Users 8, 9, and 10, who bet on Outcome 2 and Outcome 3, do not receive any votes. And User 1, the architect, is paid out their reward in votes as well.

4) Transferring Coins: A transaction in the Fortuna protocol is perhaps the most readily understood of the elements because its function is that of a transaction in a typical blockchain. They are used to transfer coins from one address to another by specifying both the recipient address and the number of coins being transferred.

IV. Fortuna Implementation

A. Staked Voting

A crucial requirement of a decentralized network of oracles to ensure trust in its users is for each oracle to have an incentive to vote honestly or a disincentive to vote dishonestly. To satisfy this requirement, Fortuna implements a unique staked voting algorithm (Fig. 4). Winning bettors are paid out in votes (one vote for each coin won) and required to place those votes on another pool's outcome in order to redeem their winnings. Each vote placed with the consensus will pay out a single coin, whereas each vote placed against the consensus will be lost. Below is a more detailed description of this process.

First, consider rounds of pools, bets, and votes. This is not a requirement in practice but allows for easy illustration. Let p_i, b_i, and v_i define the set of pools, bets, and votes in the i^{th} round. Round 1 begins with a set of pools p_1, bets b_1, and votes v_1. Now, assume the first round of pool outcomes have been determined and the network agrees. This concludes Round 1. Round 2 begins in the same way, with a set of pools, p_2, and bets, b_2. At this point, the pools are closed, and the bettors are waiting on outcome determination from the voters, v_2, to redeem their winnings.

Fortuna's method of staked voting requires the winning bettors in Round 1 to vote on a random pool in Round 2 before redeeming their winnings from Round 1. Chosen randomly, the set of voters for the j^{th} pool in Round 2, $v_{2j} \subseteq_R b_1{}' \subseteq_R b_1$ where $b_1{}'$ represents all winning bettors from Round 1. Additionally, a voter in the set v_{2j} is only given access to their Round 1 winnings if they vote with the consensus among other voters in that set. This methodology is premised on the basic concept that the easiest way to ensure consensus among strangers is with the truth.

In practice, truly randomizing the set of voters for a given pool is difficult. However, we can approximate randomness by mathematically limiting the number of pools on which a user can vote, based on a "voting partition." We discuss voting partitions further in the next section.

This method of staked voting minimizes the feasibility of an attack on the integrity of the blockchain. Requiring each user in v_{2j} to vote with the consensus in order to redeem their Round 1 winnings incentivizes each voter to vote honestly. Using this staked voting method of outcome determination, each round of voting relies on the previous round of betting and, therefore, the network will run itself as long as subsequent pools are created, and bets are placed.

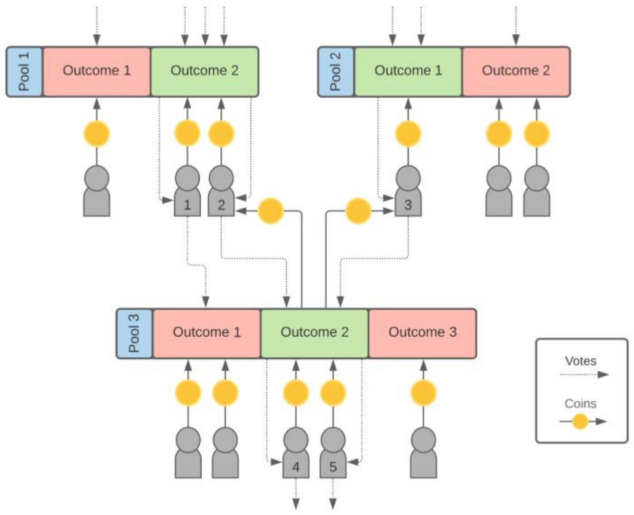

Fig. 4. This figure shows bettors and voters from three pools. For simplicity, architects were removed. Once the winning outcomes are determined for Pools 1 and 2 (by voters not shown), the winning bettors (Users 1, 2, and 3) are paid out in votes. Users 2 and 3, who placed those votes with the consensus on Pool 3, collect their winnings. User 1, who placed their votes against the consensus, forfeits their winnings. Users 4 and 5, the winning bettors from Pool 3, continue this process. Losing bettors are not permitted to vote.

B. Voting Partitions

As noted previously, winning bettors are not paid out with coins, but instead with votes. In order to redeem coins from those votes, a user must vote with the consensus on a closed pool. However, to defend against Sybil attacks, the pools on which a user can vote are randomized by using "voting partitions." Each vote and pool is assigned a random partition with the requirement that votes can be placed only on pools within their own partition.

A voting partition is defined by the leading bits of a vote or pool's "partition key." We define a vote's partition key as

$hash(bet_{id} \oplus block_{id})$ where bet_{id} is the ID of their previously won bet and $block_{id}$ is the ID of the block in which their bet was placed. And similarly, we define a pool's partition key as $hash(pool_{id} \oplus block_{id})$ where $pool_{id}$ is the ID of the pool, itself, and $block_{id}$ is the ID of the block in which the pool closed.

With this design, we can guarantee random pool assignment across voters. While bet_{id} and $pool_{id}$ are not guaranteed to be random, $block_{id}$ is. A block's ID is generated by hashing serialized block data with a random nonce in order to meet specific proof of work requirements. Because it would be impractical for a miner to generate multiple valid block IDs in the hopes of producing a desired partition key, we can assume the block ID of a given block is effectively random. Therefore, because each partition key is generated by hashing either the bet ID or pool ID XORed with a random block ID, we know the final result must also be random.

C. *Vote Capping*

While voting partitions defend against a nefarious user winning multiple bets and placing all the votes on a single pool in order to unilaterally determine the winning outcome, vote "capping" defends against a nefarious user winning a single large bet and doing the same. A user's vote contribution is limited to, at most, 20% of a single pool's quorum. This prevents a user with a large winning bet from having too much voting power over a smaller pool.

D. *Quorum*

A quorum is required to determine when voting on a pool is closed and payouts can be made. How this quorum is defined is crucial in ensuring the network runs smoothly and efficiently.

If the quorum is too small, over time, there would become a surplus of votes. Users would be unable to place their votes, preventing them from redeeming their winnings and participating on the platform. A large number of coins would be locked up in unplaced votes, causing the system to stall. On the other hand, if the quorum is too large, over time, pools would fail to reach quorum and fail to settle. The bets placed on those pools would be left unresolved and the bettors would be unable to redeem their winnings. A large number of coins would be locked up in unsettled pools, again causing the system to stall.

As a result, the number of votes required to settle a pool must be chosen carefully. To prevent a scenario from occurring where there is a surplus of unresolved bets or a surplus of unplaced votes, there must be an equal number of coins and votes in circulation at any one time. Accordingly, we must ensure a pool produces as many coins as it consumes and that it produces as many votes as it consumes. Otherwise, over time, there will either be a surplus of coins or a surplus of votes.

We can make this guarantee by setting the quorum of a pool equal to the number of votes that pool will generate. Because winning bets are paid out in votes (one vote for each coin won), a pool will generate as many votes as the total pot. Therefore, a pool's quorum must equal the total pot. In this way, as more coins are bet on a pool, more votes will be produced. And, as more votes are placed on a pool, more coins will be produced.

E. *Payout*

Once a pool has reached a quorum of votes and a winning outcome has been established, the pot is distributed among winning bettors based on (1). This equation ensures winnings are proportional to the size of a user's bet and the sum of all winnings are equal to the size of the pool's pot (minus one percent to reward the pool's architect, as represented by the constant 0.99). Let W be the number of votes rewarded from a won bet, b be the amount of the won bet, b_w be the amount of all winning bets on that pool, and p be the total size of the pool's pot. The pot includes all bets as well as the architect's initial fee.

$$W = 0.99\left(\frac{b}{b_w} \cdot p\right) \tag{1}$$

If a pool completes in a "no contest," where the majority outcome was not one provided by the architect, all bettors' initial bets will be returned. However, they still must first vote with the consensus on another pool in order to redeem their returned bets. This ensures an equilibrium of coins and votes in circulation.

V. CONCLUSION

A. *Results*

To test the protocol, a simulator was constructed wherein simulated users created and validated new actions, mined blocks, and managed balances and payouts. Each user would broadcast actions containing a random set of elements based on probabilities chosen to allow a large number of bets to be placed on a large number of pools while also incorporating a non-negligible number of malicious votes. Additionally, users were chosen randomly to mine validated actions, claiming the corresponding mining reward.

To validate the protocol, the initial value of the network was compared to the final value of the network after running the simulation. The final value was calculated with the total value of all user actions, the total value of all open bets (bets placed on pools that had not yet closed), and the total value of all unredeemed winnings (winnings that had not yet been paid out). The simulation demonstrated that expected consistency.

B. *Summary*

The objective of this work was to provide a means of bringing real-world event outcomes onto a blockchain in a reliable and trusted way. We do this by introducing Fortuna, a distributed oracle network demonstrating a novel staked voting algorithm, introducing the concept of voting partitions, and implementing a unique pari-mutuel betting design.

As impactful as decentralized blockchain technology has become since the inception of Bitcoin in 2008, there are still challenges. One challenge addressed in this paper is the ability to bring real-world event outcomes onto the blockchain. Most decentralized blockchains do not have this capability. However, there are existing oracle networks that attempt to solve this problem – ZenSports, Wagerr, and Augur. Each has its own drawbacks such as: (1) major limitations on supported event types, (2) reporting algorithms susceptible to attacks, and (3) varying degrees of centralization.

Fortuna, a distributed betting platform motivated both by the large gaming industry as well as the need for a more robust and capable oracle platform, solves these problems in a novel way. Fortuna provides players with a more affordable and favorable means to engage in gambling through its use of actions and action elements. An action contains one or more elements representing either a pool, a bet, a vote, or a transaction. Users interact with the network by creating pools representing real-world events, placing bets on those pools, casting votes on pool outcomes, and transacting with other users.

In order to incentivize users to vote honestly on event outcomes, Fortuna implements a novel staked voting algorithm. This requires winning bettors to vote with the consensus on another pool in order to redeem their winnings. If the user votes with the consensus, their winnings will be released. If, however, the user votes against the consensus, their winnings will be forfeited. To defend against a Sybil attack and ensure random pool assignment, Fortuna introduces the concept of voting partitions as well as vote-capping. Voting partitions randomize the pools on which a bettor can vote, and vote-capping limits the number of votes a bettor can place on a single pool. This defends against an attacker placing either one large bet or many small bets in the hopes of then placing a large number of dishonest votes on a particular pool.

Additionally, because there is no centralized casino against which a user can place a bet, bets are made with, and against, other users. A pool accepts bets from all sides and pays out winning bettors based on the total pot. This pari-mutuel design allows the odds of a bet to change organically based on public perception and, with enough bets, approach an equilibrium. In addition, this removes any requirement for a binary outcome. With a pari-mutuel design, there is effectively no limit on the number of potential outcomes an event can have.

This novel method of staked voting, the concept of voting partitions, and the pari-mutuel design creates a fully distributed, self-handicapping, self-regulating betting platform and allows non-binary real-world event outcomes to be determined in a reliable and trusted way by otherwise trustless users.

C. Future Work

While the system and protocol have been tested with favorable results, there are areas of improvement and further study. One opportunity for improvement is to provide a defense against coercion attacks.

A coercion attack is one where an adversary bribes a user on the network to vote in a certain way, regardless of the actual outcome of an event [18]. These types of attacks can occur on networks that provide “receipts” for a user’s vote. In many decentralized voting systems, once voting has completed, the individual votes are made public to the network. This “receipt” allows a voter to prove to an adversary whether or not they voted as requested. In order to defend against this type of attack, the individual votes must be computationally secret to the network, even after a quorum is reached and the overall vote is determined.

Work has been performed to create decentralized voting systems that defend against these attacks [18, 19]. While we have demonstrated a working system with numerous security properties to bring real-world events onto a blockchain, it would be beneficial to extend this work to encompass defense against coercion attacks.

REFERENCES

[1] Back, Adam. "Hashcash-a denial of service counter-measure." (2002).

[2] Lee Kuo Chuen, David. “Handbook of Digital Currency.” Elsevier, 2015.

[3] CoinMarketCap. “Global Charts.” Accessed October 13, 2020. https://coinmarketcap.com/.

[4] Greenspan, Gideon. “Why many smart contract use cases are simply impossible.” Accessed November 8, 2020. https://www.coindesk.com/three-smart-contract-misconceptions/.

[5] Peck, Morgen E. "Blockchains: How they work and why they'll change the world." IEEE spectrum 54.10 (2017): 26-35.

[6] Mühlberger, Roman, et al. "Foundational Oracle Patterns: Connecting Blockchain to the Off-chain World." International Conference on Business Process Management. Springer, Cham, 2020.

[7] Beniiche, Abdeljalil. "A Study of Blockchain Oracles." arXiv preprint arXiv:2004.07140 (2020).

[8] Peterson, Jack, and Joseph Krug. "Augur: a decentralized, open-source platform for prediction markets." arXiv preprint arXiv:1501.01042 (2015).

[9] “ZenSports: Peer-to-Peer Mobile Sports Betting Using Blockchain and Cryptocurrencies to Eliminate the Bookmaker.” Accessed October 13, 2020. https://www.zensports.com.

[10] Mah and Christensen. “Wagerr: The Decentralized Sportsbook of the Future Powered by Blockchain Technology.” white paper (2018). https://web.archive.org/web/20200725215353/https://www.wagerr.com/wagerr_whitepaper_v1.pdf.

[11] American Gaming Association. “State of the States 2019.” Accessed October 13, 2020. https://www.americangaming.org/resources/state-of-the-states-2019-the-aga-survey-of-the-commercial-casino-industry/.

[12] Antonopoulos, Andreas M. “Mastering Bitcoin: unlocking digital cryptocurrencies.” O'Reilly Media, Inc., 2014.

[13] Chuen, David LEE Kuo, Li Guo, and Yu Wang. "Cryptocurrency: A new investment opportunity?." The Journal of Alternative Investments 20.3 (2017): 16-40.

[14] Nakamoto, Satoshi. “Bitcoin: A peer-to-peer electronic cash system.” Manubot, 2019. https://git.dhimmel.com/bitcoin-whitepaper/.

[15] Thomas, Mark. “Creating Bets as a Maker.” Accessed November 9, 2020. https://support.zensports.com/en/articles/2651500-creating-bets-as-a-maker.

[16] Thomas, Mark. “Responsibilities, Penalties, & Rewards As A Decentralized Betting System.” Accessed November 9, 2020. https://support.zensports.com/en/articles/2986212-responsibilities-penalties-rewards-as-a-decentralized-betting-system.

[17] Ottaviani, Marco, and Peter Norman Sørensen. "Parimutuel versus fixed-odds markets." Unpublished Paper (2005). https://web2.econ.ku.dk/sorensen/papers/pvfom.pdf.

[18] Benaloh, Josh, and Dwight Tuinstra. "Receipt-free secret-ballot elections." Proceedings of the twenty-sixth annual ACM symposium on Theory of computing. 1994.

[19] Dimitriou, Tassos. "Efficient, coercion-free and universally verifiable blockchain-based voting." Computer Networks 174 (2020): 107234.