SportCrypt

A decentralized prediction market built on the ethereum blockchain

Version 0.3

December 2017

https://sportcrypt.com

Contents

1 Introduction

The designers of a prediction market must make many design and implementation decisions that will determine the ultimate utility of the platform. With SportCrypt we believe we have developed an excellent balance between simplicity, efficiency, and features. This paper explains the design and implementation of the SportCrypt platform at a technical level.

1.1 Sports Betting

Although the SportCrypt contract is not specific to sports, this is our initial area of focus. For our purposes, sporting events have the following advantages over other instruments:

- Sports betting is a massive existing market that is not well-served by existing systems. Current systems suffer from unreliable, slow, and expensive payment processing, jurisdictional roadblocks, high trading fees, and significant counter-party risk. We believe that all of these problems have solutions in a distributed application.
- Most sporting event outcomes are objective truths. These truths can by verified by watching the events on television, or by checking officially posted scores. Because of this, distributed oracles don't appear to be as necessary as they are in other applications. There is less of an incentive for oracle manipulation since any alteration or misinterpretation of official results will be immediately obvious, to the detriment of the exchange's reputation.

1.2 ETH-Denominated

In SportCrypt, all trading is done with ether (ETH). No other tokens are necessary.

Since ETH is by far the most liquid asset on the ethereum network, and is already a prerequisite for incorporating transactions into the ethereum blockchain, we feel that adding new special-purpose tokens to the trading flow is undesirable. Not only would users need to learn how and where to acquire these tokens, but to do so they would need to pay market spreads (which will be high prior to substantial demand) and trading fees (which may be denominated in yet more special-purpose tokens).

Granted, even obtaining and using ETH is an obstacle for non-technical users. However, compared to other demographics, sports bettors are accustomed to having to jump through various hoops prior to being able to place bets. There is in fact an entire industry built around moving funds in unorthodox ways for the purpose of sports betting (NetTeller, Skrill).

1.3 Custody of Funds

The primary advantage of using a smart contract for sports betting is to remove counter-party risk for the users of the exchange. Assuming there are no bugs in the

smart contract, not even the operators of the exchange are able to access funds held in balance or locked into trades. Nor are they able to freeze funds and accounts to prevent trading or withdrawals. The only attack vector for a malicious exchange is to mis-report the outcome of an event, in which case all participants of the match will be affected, and there will be indisputable evidence of the misbehavior recorded on the blockchain. We will discuss this further in the Oracles section of this paper.

If for whatever reason SportCrypt disappears or doesn't finalize a match, funds can be recovered after waiting a certain period of time. See the Funds Recovery section below.

A corollary to SportCrypt not being able to freeze funds or interfere with trading is that we have no ability to reverse or unwind any activity on our platform. All trades are final. However, we are committed to building a reputable business. Any users who feel they have suffered losses due to errors on our part are encouraged to contact us.

1.4 Fees

Our initial plan is to charge no fees or commissions. The only fees needed to participate in SportCrypt are the gas fees required by the ethereum network. See the Efficiency section below for an approximate quantification of gas costs.

Since the exchange deducts no fees or commissions, the trades placed through SportCrypt are entirely zero-sum between peer users of the exchange. This is hypothesized to have positive legal ramifications. We believe that a trade placed through SportCrypt is of the same nature as two individuals making a private bet between themselves.

1.5 Fixed-odds

All trades on SportCrypt are **fixed-odds** bets. This means that the odds of a trade are known and agreed upon beforehand by the parties involved, and they cannot be changed after the fact. Of course, new subsequent trades can be made at different odds. Making new trades in the same direction as previous trades can serve to average up or down the position's cost basis, and trades in the opposite direction can either fully or partially close out a position at a profit or a loss, in which case that account's balance will be immediately credited.

Fixed-odds betting is distinct from parimutuel betting where all the bets on an event are pooled together and therefore the odds are only known at the end of the betting session (usually right before the event starts).

1.6 In-game Trading

Orders may be offered and trades may be created at any point up until match finalization. This allows users to enter into new positions or exit existing positions at any point prior to or during an event. Users who are unwilling to wait for finalization may even trade after the event has completed so as to have their balance available immediately (at a small cost).

We view this as a very important aspect of SportCrypt. Being able to trade at half-time and TV intermissions, or even during live game play, adds a new level of excitement to the trading experience. However, due to the nature of distributed consensus as implemented by ethereum, certain considerations need to be kept in mind. When two or more conflicting transactions are broadcast to the network, it is indeterminate which one will be mined first. For example, if an unexpected play occurs, a market maker who has an outstanding order may attempt to cancel the order at the same time one or more participants attempt to trade on it. Whether a trade is made, and by whom, is indeterminate.

Furthermore, not only is transaction ordering indeterminate, it can also be influenced by gas price which adds a new dimension to in-game trading, one that may be attractive to sophisticated traders. Participants who don't wish to include gas prices in their trading models are advised to restrict in-game trading to half-time, timeouts, TV intermissions, etc, and to make careful use of the order expiry parameter.

1.7 Partial Trades

Orders may offer any amount of ETH to be traded, and orders may be completely or partially filled by trades created by one or more users. Once a trade is created, each of the two parties to the trade will have **positions** on the event, meaning that one of the parties will profit given one outcome, and the other will profit given the opposite. However, if either party wishes to exit their position, and there are willing participants in the market, the position may be fully or partially sold off, either at a different price (for a profit or a loss), or at the same price (for no profit or loss).

1.8 Collateral

In order to enter trades, users need to have balances in the SportCrypt contract. These balances are tied to ethereum public keys and the smart contract ensures that only the corresponding users can choose whether their balances are withdrawn or traded.

Collateral requirements on SportCrypt are more flexible than on many other platforms, and are designed with market makers in mind. Creating an order does not reserve any funds from a user's balance. In fact, a user may create many orders all backed by the same account balance. Only once a trade is executed are funds reserved. If this reduces the balance enough that it affects the ability to fill the other outstanding orders, those orders are automatically reduced or cancelled to compensate.

Because users only need a balance to enter into trades, they can in fact create orders when they have 0 balance. The orders won't be visible to anyone until the user deposits funds, claims winnings from a finalized match, or closes an existing position in a different match. At that point the orders will appear in the order-book.

Additionally, trades made in the opposite direction of an existing position can use the position as collateral. Because of this, given a position on an event and a empty account balance, orders can still be created that, if filled, will partially or fully close out that position. And furthermore, the proceeds from closing that position can be used to create a new opposite position, even within the same trade.

2 Matches

Prior to trading a match, its **match details** should be examined. These details represent the rules of this particular match and should be fully understood prior to trading.

2.1 Match Details

When matches are posted on SportCrypt, they are posted with corresponding UTF-8 encoded JSON values which contain the match details. Here is an example:

```
{
    "type": "sports/nfl/game",
    "event": {
        "kickoff": "1509296400",
        "spread": "-2.5",
        "team1": "OAK",
        "team2": "BUF"
    },
    "contractAddr": "b9fea0142cd54bd0a8238cba4a286f5a1a261692",
    "nonce": "ul36TwZFutiL9nTlpHkMV5",
    "cancelPrice": "50",
    "recoveryWeeks": "12"
}
```

- **type**: The type of match as a hierarchical classification, separated by `/` characters.
- **event**: A nested object containing fields specific to the `type` of this match. For example, `sports/nfl/game` has the following parameters:
  - **kickoff**: The unix timestamp of when the match is scheduled to begin.
  - **name**: The special name of the match, for example "Super Bowl" (optional).
  - **spread**: The point spread for this match (explained below).
  - **team1**: The short-form identifier for the visiting-team of this match.
  - **team2**: The short-form identifier for the home-team of this match.
- **contractAddr**: The SportCrypt contract address this match will be traded on.
- **nonce**: A random value that prevents the prediction of match IDs, and permits the creation of otherwise identical matches.
- **cancelPrice**: If the outcome of a match cannot be determined, usually because the match was called off, this is the price the match will be finalized at (normally by the exchange, but potentially instead by funds recovery).
- **recoveryWeeks**: Weeks after the first trade that must elapse with no finalization before funds can be recovered at `cancelPrice`.

All values are strings (even numeric fields like point-spreads, prices, timestamps, and weeks). Ethereum addresses are stored in lower-case hex, without the `0x` prefix.

2.2 Match IDs

In order to compute the 256-bit **match ID**, the match details JSON is sorted alphabetically by key, minified (all unneeded white-space characters are removed), and then hashed with the `keccak256` cryptographic hash function.

The resulting hash is then truncated by 2 bytes and is appended with 2 `uint8` bytes: the first byte encodes the `cancelPrice` parameter of the match, and the second encodes the `recoveryWeeks` parameter.

The resulting value is the match ID which is used on the blockchain and in signed orders.

When displayed, it is usually expressed as a hexadecimal string, for example:

`8c1705e212fd2d369e57e0012fa1e3083705cd71a871af21f6ce6230cfcd320c`

The first 60 characters represent the 30-byte truncated hash of the match details. The following `32` byte represents a `cancelPrice` of `50`, and the final `0c` byte represents a `recoveryWeeks` of `12`.

Prior to creating orders or trades for a match, users should verify that the match ID is computed correctly from the match details. Our reference UI implementation does this automatically.

2.3 Point Spreads

Depending on the match type, it may have an associated **point spread**. This is a positive or negative number that is added to the final score of `team2` (the home team) prior to evaluating the outcome of a match. This is done so that even teams with different skill levels can be traded at close to even odds, and also so that ties ("pushes") cannot occur (because in most sports scores are integers but point spreads have fractional components). SportCrypt chooses its point-spreads based off the initially posted vegas or off-shore lines.

Using the above example, matches have short names such as "OAK@BUF-2.5". This is short for a contest where the Oakland Raiders play against the Buffalo Bills. The "@" sign indicates that Oakland is the visiting team and Buffalo is the home team. The "-2.5" indicates the point spread that is to be applied to the home team's score. Since in this case it is negative, it is disadvantaging the home team, meaning that, not considering point spreads, Buffalo is considered more likely to win this match.

In SportCrypt, the contract **postulate** is always about whether the visiting team will win against the point spread. So, in order to determine the outcome of this contract, add -2.5 to Buffalo's final score and compare that against Oakland's final score. If Oakland's is greater then the outcome has occurred (the postulate was true), otherwise it hasn't (it was false).

3 Prices and Odds

Understanding pricing and odds is critical to profitably trading on SportCrypt (and elsewhere). Simultaneously buying and selling at different prices/odds is how market makers earn profit, and accurately assessing the probability of events and comparing those to posted prices (or creating their own prices) is how traders earn profit.

On SportCrypt anyone can be a market maker, a trader, or both.

3.1 Implied Probability

On SportCrypt, each contract is valued at an integer **price** from 0 to 100. This corresponds to the **odds** in traditional sports betting, since it reflects the perceived chances that an outcome will occur, and therefore the amount that a trader will need to risk to earn a given amount.

When it is finalized, either the contract postulate will have been found to be true, in which case the contract will be finalized at a price of 100, or it will have been found to be false, in which case it will be finalized at a price of 0. Prior to finalization, market participants can choose to value a contract at prices in the range between 0 and 100.

This range from 0 to 100 is chosen so as to map to probabilities expressed in percentage. For this reason, odds in this format are called **implied probability** odds. The most significant popularizer of implied probability in sports betting was the now-defunct website TradeSports.com, so in this paper we refer to implied probability pricing of binary options as the "TradeSports model".

3.2 Bid-Ask Spread

The difference between the lowest ask and the highest bid is called the **bid-ask spread**. This spread is unrelated to the point spread discussed previously.

Because market makers attempt to buy at low prices and sell at high prices, they prefer large bid-ask spreads. Conversely, because they must pay market prices, traders who execute trades prefer small bid-ask spreads.

In a popular and competitively traded event, the bid-ask spread is typically smaller ("tighter") than in an unpopular event. This is because market makers tend to compete with each other by offering smaller bid-ask spreads, and also because traders will create orders at slighty better prices than the market makers, hoping to avoid paying the bid-ask spread to a market maker by selling directly to another trader.

In a centralized exchange, it is usually impossible to have a negative bid-ask spread (where a bid is at a higher price than an ask) because these overlapping orders would be filled immediately. However, in SportCrypt negative bid-ask spreads are possible since the order-book is decoupled from execution. Negative bid-ask spreads should be uncommon though because they represent opportunities for arbitrage. This is where an opportunistic trader simultaneously buys at the low ask price and sells at the high bid price so as to profit from the difference. A useful feature of ethereum allows this to happen atomically: Either both of the trades will execute or neither of them will.

As well as negative bid-ask spreads, a match on SportCrypt can also have a bid-ask spread of zero. In this case there is no opportunity for arbitrage and the bid-ask spread can be thought of as the gas required to execute a trade.

3.3 Amount at Risk

When creating a trade, participants agree upon an integer price $Price_{trade}$ between 0 and 100 (non-inclusive). The amounts they must put at risk to form the trade depend upon this price, and are defined by this formula:

$$A_{buyer} = A_{seller} \times \frac{Price_{trade}}{100 - Price_{trade}}$$

Or equivalently:

$$A_{seller} = A_{buyer} \times \frac{100 - Price_{trade}}{Price_{trade}}$$

The total trade amount is simply the sum of the two amounts at risk:

$$A_{total} = A_{buyer} + A_{seller}$$

3.4 Finalization Prices

When a match has finished, a finalization price $Price_{final}$ will be published. This means the buyer can claim the following amount:

$$A_{total} \times \frac{Price_{final}}{100}$$

And the seller can claim:

$$A_{total} \times \frac{100 - Price_{final}}{100}$$

Finalization prices are usually either 100 or 0. This means that either the buyer or the seller respectively will be able to claim the entire amount, and the other will be able to claim nothing.

However, in certain rare circumstances a match will have no determinable outcome and will need to be finalized at a **cancel price**. The cancel price is a term in the match's contract details and should be accounted for in trader models. Typically the cancel price will be `50`, however a different price may be provided if the initial market value of a contract is anticipated to be materially different. This may be the case for "money-line" matches (matches without point-spreads).

3.5 Expected Value

With implied probability, it is immediately evident that it is preferable to buy at low prices and sell at high prices.

Given an accurate estimate of the probability of an outcome occurring, there will be a positive expectation when buying at prices below this estimate or selling at prices above it.

Specifically, the expected value of a buy-side trade is the following (where $P_{estimate}$ is a probability, not a price):

$$E = (P_{estimate} \times A_{seller}) - ((1 - P_{estimate}) \times A_{buyer})$$

And similarly, the sell-side:

$$E = (P_{estimate} \times A_{buyer}) - ((1 - P_{estimate}) \times A_{seller})$$

Using expected value, standard bankroll management techniques such as the kelly criterion can be applied.

3.6 Odds Conversion Examples

Implied probability has several advantages over other odds representations. However, current sports bettors are familiar with a variety of formats so we will next present some abridged odds conversion tables:

Buy-side

Implied Probability	American	Decimal	Fractional
10	+900	10	9-1
25	+300	4	3-1
33	+203	3.03	~2-1
40	+150	2.5	3-2
50	+/-100	2	1-1
60	-150	1.67	2-3
66	-194	1.52	~1-2
75	-300	1.33	1-3
90	-900	1.11	1-9

Sell-side

Implied Probability	American	Decimal	Fractional
10	-900	1.11	1-9
25	-300	1.33	1-3
33	-203	1.52	~1-2
40	-150	1.67	2-3
50	+/-100	2	1-1
60	+150	2.5	3-2
66	+203	3.03	~2-1
75	+300	4	3-1
90	+900	10	9-1

4 Off-Chain Mechanics

SportCrypt uses a hybrid on/off-chain approach. Match details, match finalization prices, and orders are communicated off-chain, but trade settlement occurs on-chain. We refer to this as the "EtherDelta model".

We are also evaluating a modification to this model where matching happens centrally and is settled on-chain by transactions sent by the exchange. We call this the "IDEX model", although we won't discuss it further in this paper.

4.1 Match Creation

When new matches are offered for trading, they are added to our backend system which forwards them to all connected clients via websocket. The exchange doesn't need to perform any on-chain actions to create a new match. This is a major scalability advantage of SportCrypt since it reduces the operational costs of the exchange and allows us to experiment with many simultaneous matches without worrying about ethereum gas overhead or waiting for blocks to be mined.

4.2 Match Finalization

Similarly, finalizing a match requires no on-chain actions. The operators of the exchange will sign a message indicating that a given match ID is to be finalized at a certain price. Similar to orders, the signature is specific to a certain contract address. When claiming their winnings, participants (if there were any) will submit this signed message, along with the signature, to the blockchain. If nobody else has finalized the contract yet, the signature will be validated and the contract will be finalized. The consequence is that the first participant to claim their winnings will pay an extra 10k in gas. However, we feel this is worth it because it relieves the exchange from any operational overhead for finalizing matches.

4.3 Orders

After selecting a match to trade, market participants create and sign orders which are then uploaded to our off-chain order-book via websocket.

The orders are tightly packed into 3 `uint256` values:

```
[0]: 32-byte match ID
[1]: 32-byte amount in wei
[2]:  5-byte expiry
      5-byte nonce
      1-byte price
      1-byte direction
     20-byte address
```

- **match ID**: The hashed match details, as described above.
- **amount**: The maximum total amount at risk authorized by the order creator.
- **expiry**: Unix timestamp after which the order ceases to be valid.
- **nonce**: A random value which allows multiple otherwise identical orders to be issued, and prevents order ID prediction.
- **price**: A value from 1-99 representing the price authorized by the order creator.
- **direction**: Either 0 for a sell order, or 1 for a buy order.
- **address**: The address of the order creator.

Packing is necessary to work around the solidity stack depth limitation, as well as to save on calldata gas expenses. SportCrypt's 96 byte orders are quite lean (compared to EtherDelta's 136 bytes and 0x's 292 bytes).

The **order ID** is the `keccak256` hash of the contract address concatenated with these 3 `uint256` values.

After being submitted to the order-book, orders and their signatures are propagated to all connected clients who have subscribed to the given match.

When a market participant chooses to execute a trade, the order and its signature are sent to the blockchain to be executed. The smart contract computes the order ID and verifies that the order creator's address matches the provided signature.

5 On-Chain Mechanics

This section is an outline of the smart contract's core trading mechanics expressed in an idealized mathematical form. The actual implementation is structured slightly differently so as to avoid rounding loss, enable efficient invariant assertions, and to optimize gas usage. This description should not be taken as authoritative; the authoritative description is the smart contract source code.

5.1 Trading

The `trade` function accepts an order, the signature of the order creator, and an amount. All of the trade details are determined by the order parameters, except the amount. This amount is the largest amount at risk the trader is willing to accept, and the smart contract will attempt to create a trade with an at risk component as large as possible up to that amount.

Under normal operation, the smart contract is designed to never `throw` exceptions since this has the undesirable effect of consuming all provided gas. Instead, the trade may complete with a 0 amount, in which case a `LogTradeError` log message may be issued. See the `Status` enum in the smart contract for details on how to interpret these log messages.

After a match has been finalized, trading for this match is disallowed by the smart contract

5.2 Positions

Positions are the result of trading, and they maintain the state of which accounts are eligible to collect the reserved funds once contracts have been finalized. Negative values for positions represent short positions, and positive values represent long positions. The positions represent the total amounts to be claimed, not the long or short amounts at risk.

5.3 Effective Balances

To create a trade, users normally require sufficient funds in their unallocated account balance, $Bal_{account}$, to match their amount at risk. However, if a trade is made for a match the user already has a position on, and the trade is in the opposite direction of the existing position, then the position itself may be used as collateral. In this way, it is always possible to create new trades to close out existing positions.

Here is how the effective balance is computed for a buyer:

$$Bal_{effective} = Bal_{account} + \begin{cases} -Position \times \frac{Price}{100} & \text{if } Position < 0 \\ 0 & \text{if } Position >= 0 \end{cases}$$

And for a seller:

$$Bal_{effective} = Bal_{account} + \begin{cases} Position \times \frac{100 - Price}{100} & \text{if } Position > 0 \\ 0 & \text{if } Position <= 0 \end{cases}$$

$Bal_{effective}$ is computed for both parties to a trade. These values are then used to determine the amounts that will be used in the trade, along with the remaining order amount, the maximum trade amount, and the order price.

5.4 Position Updates

When a trade is created, the positions for the participating accounts are updated by adding or substracting the total trade amount, A_{total}, as follows.

For the account performing the buy side of the trade:

$$Position_{new} = Position_{old} + A_{total}$$

And for the account performing the sell side:

$$Position_{new} = Position_{old} - A_{total}$$

Since there are opposite positions of equal magnitude created for every trade, prior to finalization all positions on a match net to 0:

$$\sum_i Position(account_i) = 0$$

After finalization this invariant no longer holds because winners will claim their positions and in the process reset them to 0. Participants on the losing side have no reason to pay the gas to set their positions to 0.

5.5 Balance Updates

As well as updating the positions of the participating accounts, the balances of the accounts are modified according to the following formulae.

For the account performing the buy side of the trade:

$$\begin{aligned} Bal_{new} = Bal_{old} \\ &+ ((-Position_{old} + min(0, Position_{new})) \times \frac{100 - Price}{100}) \\ &- ((Position_{new} - max(0, Position_{old})) \times \frac{Price}{100}) \end{aligned}$$

And the sell side:

$$\begin{aligned} Bal_{new} = Bal_{old} \\ &+ ((Position_{old} - max(0, Position_{new})) \times \frac{Price}{100}) \\ &- ((-Position_{new} + min(0, Position_{old})) \times \frac{100 - Price}{100}) \end{aligned}$$

The intuition behind these equations is that an account's balance is debited for increasing the magnitude of a position and credited for reducing it. Since a position is being sold or purchased, the debit or credit amount depends on the price agreed upon for the trade.

If there are no preexisting positions, then both balances are decreased by the corresponding amounts necessary to cover the respective amounts at risk needed for the trade. However, if one or both of the parties have existing positions, and one or both of them are trading in an opposite direction to their positions, then the balances may increase. This is not always the case though since a single trade can close out an existing position (increasing balance) and additionally create an opposite position (decreasing balance).

5.6 Order Amount Decrease

When a trade is made, a filled amount variable for the corresponding order ID is incremented by the amount at risk that was consumed by the trade.

The difference between the filled amount and the order amount is what is used for later trades when determining how much of an order remains.

When cancelling an order, the filled amount is increased to be the same as the order amount (see the next section).

5.7 Order Cancellation

Similar to EtherDelta, order cancellation must be done on-chain by calling a `cancelOrder` method of the smart contract. Unfortunately this means that to cancel an order, gas needs to be paid and the cancellation is not instantaneous.

This can be avoided in many cases by short-term order expiry values. Once `block.timestamp` has exceeded the order expiry timestamp, the order will be effectively cancelled and there is no need to cancel it with a transaction.

5.8 Funds Recovery

As explained in the Match ID section, the `cancelPrice` and `recoveryWeeks` parameters are available to the smart contract because they are embedded in the match ID. This otherwise undesirable in-band signaling and reduction of hash strength is necessary to implement the funds recovery feature described in this section.

Note that we are truncating the 256-bit `keccak256` algorithm by 16 bits, leaving 240 bits of digest, still a very comfortable security margin ($\sqrt{2^{240}} = 2^{120}$).

If the SportCrypt exchange does not finalize a match, users will need to wait until `recoveryWeeks` weeks after the first trade on that match has elapsed. At that point, any user may call the `recoverFunds` method of the smart contract. The smart contract will finalize the match at the `cancelPrice`, allowing users to claim their funds. Weeks were chosen since the 256 possible values give us good granularity and a range up to approximately 5 years. Compared to other periods of time such as months, weeks are uniformly 7 days long and have survived calendar modification since the year 46 BC.

Naturally, we hope this functionality is never used. It serves as an assurance to users that even if SportCrypt were to disappear completely, funds would still be recoverable at the `cancelPrice` that was agreed upon in the match details.

6 Efficiency

SportCrypt has been designed from the start to be a very lean project. By reducing operational costs, we are able to provide significant savings to our users.

6.1 Smart Contract

As discussed previously, the SportCrypt system is designed to do as much as possible off-chain:

- Match creation and finalization is done off-chain.
- Orders are created and advertised off-chain.

And the operations that are done on chain we've tried to keep as inexpensive as possible:

- There are no loops in contract code (except read-only views).
- Orders are tightly packed to reduce calldata gas consumption.
- The contract never throws in normal operation.

6.1.1 Approximate Gas Costs

Activity	Approximate Gas
First trade on a match	140k
Trades with no existing position	105k
Trades with an existing position	90k
Failed trade	40k
Cancelling an unfilled order	60k
Cancelling a partially-filled order	45k
First claim of winnings	40k
Subsequent claims	30k

Although the gas used is fixed, what users actually pay is determined by the gas prices they send with their transactions.

As an example, the most expensive operation requires around 140k gas. Assuming a gas price of 1 GWei, this transaction would require 0.00014 ETH. At an exchange rate of 300 USD per ETH, this transaction would cost USD $0.042 (4.2 cents).

Since gas charges are independent of trade amounts, in terms of percentage this fee can be amortized as low as desired. In this example, even a trade as small as USD $5 requires a gas fee of less than 1%.

6.2 Order-Book

Although the critical record of balances, trades, order fills, and positions are kept on the ethereum blockchain, we still need to operate servers, most importantly to provide match details and an order-book.

We have attempted to minimize our operational costs as much as possible:

- Hybrid multi-threaded/non-blocking C++ implementation.
- Communicates with clients using compressed websocket push to minimize latency and bandwidth usage.
- In-process, memory-mapped DB to store persistent data.

6.3 User Interface

The final portion of code we consider the efficiency of is the code running in the browsers of our users. In our UI we strive to use as little memory, CPU, and bandwidth as possible:

- Modern ES6+, React, Webpack code-base.
- Querying the smart contract with constant calls amortizes RPC overhead by batching requests.

Our user interface is not required to interact with SportCrypt, and we plan to release alternate UIs such as a mobile client and a command-line application.

7 Oracles

As mentioned in the introduction, SportCrypt currently implements a centralized oracle. Although some have argued that there is a path towards decentralized oracles, to us it is not yet clear whether these systems will work and, if so, when they will be trusted enough to rely upon.

The nice property of sporting events for prediction markets is that it is straightforward for honest parties to agree on the outcome of a match, either by watching the game themselves or by relying upon officially published scores. General prediction markets suffer from what we refer to as the "North Korean Missile Problem" (or the "Gore-Bush Election Problem"). On Intrade.com, there was a contract about whether North Korea would launch a missile during a certain time period. They did. This was widely reported and acknowledged as true by every expert in the field. However, the fine-print of the contract required this to be confirmed by a US government press release and there was never any press release issued. This left Intrade in the awkward position of choosing whether to violate their own posted contract rules, or to finalize a contract at a generally acknowledged as incorrect price.

The NK Missile Problem affects both centralized and distributed oracles, and the solution is not clear. With a centralized oracle, the oracle itself can unilaterally decide on the outcome, or it can defer to some other party. For instance, in the NK Missile example, Intrade deferred the decision to the US government. On the other hand, distributed oracles attempt to defer to the "wisdom of the crowds" in some manner, perhaps using economic incentives to encourage people to report what they believe to be true. However, in our view these systems remain theoretical and unproven.

In SportCrypt, we will attempt to build a reputation as a reliable oracle. If we mis-report events, users will have indisputable evidence of this because it is permanently embedded on the blockchain. Additionally, there is no way for us to report different outcomes to specific users: A mis-reported finalization price will affect every participant of the match.

That being said, should distributed oracles prove themselves to be reliable, punctual, and cost-effective, we are open to the possibility of integrating them into the SportCrypt platform.

Finally, due to the architecture of SportCrypt, it is possible for us to support "multi-sig" match finalization where two or more trusted oracles must agree upon a finalization price prior to the match being finalized. We will investigate this at a later date if the market demands higher oracle assurances.

8 Conclusion

We believe all the pieces are finally in place for a decentralized prediction market for sporting events:

- TradeSports has proven the viability of a sports betting platform modeled on a financial exchange.
- Bitcoin has spread the notion of a blockchain as a trustless, world-wide, unstoppable currency.
- Ethereum has created a blockchain implementation that allows the distribution of funds according to custom, inviolable rules.

With SportCrypt we have combined our passion for trading sporting events with decades of experience in financial markets and high-efficiency software development.

As market makers, traders, and sports fans, we've created the system that we've always wished existed, and we're inviting the world to come join us.