ChessFish V1 Whitepaper [DRAFT]

November 2023

Alexander John Lee

alexanderjohnlee@proton.me

github.com/partylikeits1983

Abstract

ChessFish is a non-custodial chess wager smart contract and chess move verification algorithm implemented for the Ethereum Virtual Machine. ChessFish v1 offers the ability for users to create chess matches that bet on the outcome of a series of chess games.

Users have the ability to specify different parameters for a wager, i.e. the ERC-20 token to wager, the number of games, and the time limit. Alongside the ability to create 1v1 chess wagers between players, the ChessFish v1 contracts allow users to create Round Robin style tournaments of players from 3 to 25 players.

ChessFish is a decentralized intermediary and chess arbiter for single chess matches as well as tournament matches. Given the decentralized nature of the protocol, payments to users of 1v1 matches or tournaments are handled by the ChessFish smart contracts. Each move in a chess game is verified by the smart contracts and the resulting status of the game i.e., checkmate, stalemate, or timeout, determines how the chess wager is settled.

Introduction

ChessFish is a smart contract that handles the payout of wagers based on the outcome of chess matches between two or more players. The validity of each move on the chessboard is verified inside of the MoveVerification contract on-chain. After each move, the smart contract checks if the game state is either a stalemate or checkmate. If the game state results in a checkmate, the game score is incremented for the winning player. If the game state results in a stalemate, the game score is incremented for both players. The player with the majority of wins is able to withdraw their wager deposit and the wager amount deposited by the other player.

After the outcome of the wager is finalized, the ChessFish contract charges a protocol fee which is sent to the PaymentSplitter contract. Token holders of the ChessFish token are able to withdraw the accumulated fees sent to the PaymentSplitter contract proportional to the number of tokens they own. The purpose of this is to give intrinsic value to the ChessFish token and for the ChessFish token to act as a dividend-paying token.

The winner of the tournament receives a ChessFish ERC-721 non-fungible token (NFT). When creating a chess wager, users can set the opponent's address to the zero address. By doing so, any other player can accept the terms of the chess wager. It is also possible for players to play against one another without betting an ERC-20 token. This is achieved when the user creating the wager sets the wager token amount to zero.

ChessFish Protocol Architecture

It is possible to represent any state of an 8x8 chess board as a single uint256 number (gameState). A chess move is encoded as a uint16 integer where the first 8 bits are the starting square coordinate and the last 8 bits are coordinates of the square the piece will move to. The two 8-bit uint8 numbers are concatenated together to form the 16-bit uint16 move value. The coordinates of the squares on the chess board are denoted in algebraic chess notation where the square 0 (0,0) corresponds to the square a1 and the square 63 (7,7) is the square h8 in standard chess notation.

White pieces on the board are represented as numbers 1-8, black pieces are represented as numbers 9-18, and empty squares are represented as the number 0. After each move, the uint256 gameState is updated in the smart contract storage. Each uint16 move is pushed to an array and saved in storage in order to check the validity of future game moves. The reason for this is that the history of previous moves is imperative for validating future moves in chess. For example, validating en passant moves as well as castling requires taking into account previous moves made throughout the game.

Each wager is represented by a unique address generated from the keccak256 hash of the addresses of both players, the address of the wager token, the time limit of the wager, the number of games, the chainId, and the blockhash at the time of the creation of the wager. If a user fails to play a move in the allotted time agreed upon in the wager, they will lose the wager, allowing the other player to withdraw the other player's deposited ERC-20 token.

ERC-20 token deposits and sends are handled by the ChessWager and Tournament smart contracts. There are two ways to make moves in a chess game using the ChessFish smart contracts: 1) by writing each move directly to the smart contract 2) by saving signed move data to an external server. Writing moves directly to the smart contract is the most secure way of ensuring a tamper proof chess game between two parties. However, doing so requires the user to send a transaction fee each time they submit a move which is potentially cost prohibitive. The alternative is for the user to sign the data using their private key and store this signed move data to a server. This signed move data can be passed to the smart contract at the end of the game where it validates that the users of the chess match made those particular moves. To remove the possibility to replay signed data, each move is unique, sequential, and has an expiration time. The ChessFish smart contract checks that signed moves are sequential, the addresses of the signed move data match the addresses of the players, and that the moves have not expired. Signing move data and storing it off chain makes it possible to use the ChessFish contracts on Ethereum mainnet.

Tournament Functionality

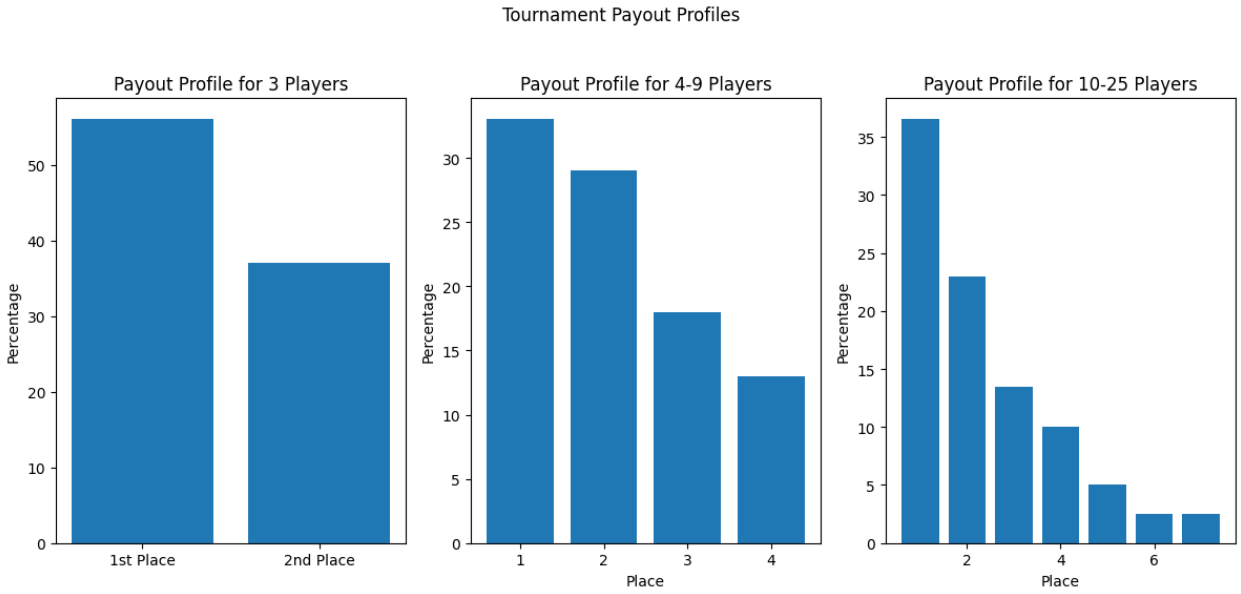
The ChessFish chess tournament functionality is handled by the Tournament contract. Tournaments are organized in a Round Robin tournament style where each player in the tournament will play at least once with each other player in the tournament. Payouts and ERC-20 interaction in tournaments are handled by the Tournament contract as opposed to the Wager contract as is the case in 1v1 matches. The number of players in a single tournament is bounded by $3 <= n <= 25$, where n is the number of players. This is to ensure an upper bound for loop iterations in the Tournament contract. The number of matches that will be played given a number of players in a

Round Robin style tournament follows the formula: $g = \frac{n\,(n-1)}{2}$, where g is the number of games, and n is the number of players.

In the Tournament contract, it is possible for players to create tournaments where each player plays every other player more than once. This means that if the number of games between each player (numberOfGames) is greater than 1, the formula for calculating the number of games in a Round Robin tournament is the following: $g = \frac{n\,(n-1)\,*\,x}{2}$, where x is the number of games between each player.

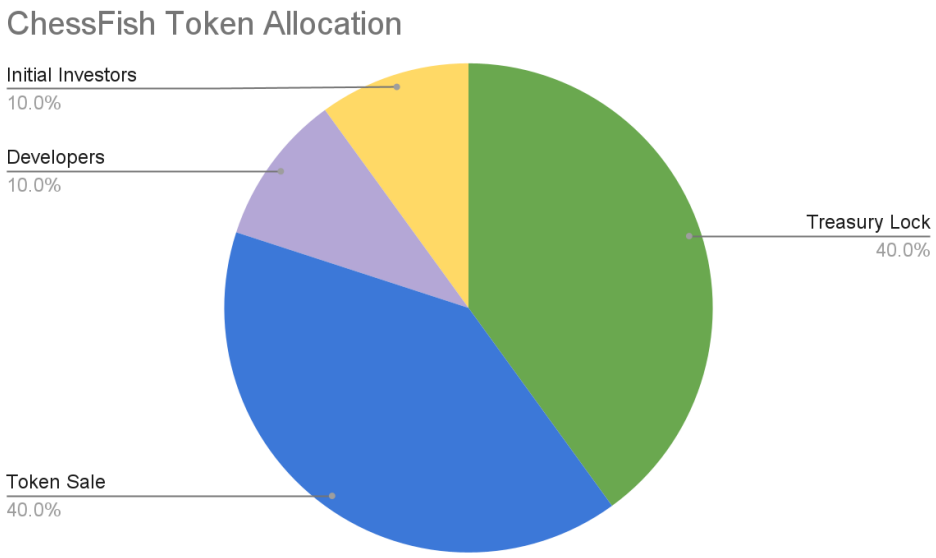
The payout amount per player in a tournament is determined by the number of players in the tournament, the tournament entry fee, and the payout profile of the given tournament. The payout profile is determined by the number of players in the tournament. The payout profile for tournaments with exactly 3 players is 56% for 1st place, 37% for 2nd place, and 0% for 3rd. This means that given a tournament pool size of 30 ERC-20 tokens, 1st place receives 16.8 tokens, 2nd place receives 11.1 tokens, and 3rd place receives 0 tokens.

The payout profile for tournaments with the number of players greater than 3 and less than 9 is 33%, 29%, 18%, and 13% for 1st, 2nd, 3rd, and 4th places. The payout profile for tournaments with the number of players greater than 9 and less than 25 is 36.5%, 23%, 13.5%, 10%, 5%, 2.5%, and 2.5% for 1st, 2nd, 3rd, 4th, 5th, 6th, and 7th places. In future versions of ChessFish, the payment profiles will be set by the ChessFish community via the ChessFish DAO. For each tournament, there is a 7% protocol fee which is sent to the PaymentSplitter contract. The PaymentSplitter contract enables ChessFish token holders to be able to withdraw accumulated protocol fees proportional to the amount of ChessFish tokens they own.



Economic Model

The ChessFish token will be initially deployed on Ethereum with a fixed supply of 1 million tokens. ChessFish token holders will be able to withdraw generated protocol fees proportionally to the number of tokens they own. 40% of ChessFish tokens will be available for sale to the community on Arbitrum after the launch of the protocol.



The dividend-paying functionality of the ChessFish token is handled by the PaymentSplitter contract. The PaymentSplitter contract is designed separately from the ChessFish token. The purpose of this is so that future versions of the ChessFish protocol can send protocol fees to newer versions of the payment splitter contracts without having to redeploy the ChessFish token.

Conclusion

The purpose of ChessFish is to serve as a neutral intermediary between two or more parties wishing to play chess while betting cryptocurrency on the outcome. ChessFish is an experimental project and has yet to be audited. Users are encouraged to exercise caution and conduct their own research before participating, given the nascent stage and potential risks associated with the protocol. As the ecosystem continues to evolve, the team behind ChessFish remains committed to refining its offerings, prioritizing transparency, and fostering a community that enjoys both the strategy of chess and the dynamics of decentralized finance.

ChessFish v1-core smart contract repository

https://github.com/Chess-Fish/v1-core