Hyper Squid Game
www.hypersquidgame.com

Hyper Squid Game Whitepaper

Abstract

Hyper Squid Game is the first decentralized lottery pool jackpot on the blockchain inspired by the original "Squid Game" series [3] on Netflix. The game itself is fully automated with permissionless functionality within the official smart contract deployed on Arbitrum, the Ethereum layer 2 blockchain used by HyperLiquid. A front-end interface for interacting with the game's smart contract has been developed on the Hyper Squid Game website [2] that allows players to easily participate in active games and obtain information regarding game progress and variable statistics.

Keywords: Smart contract, Pool, Entries, Permissionless

1. Introduction

According to the National Association of State and Provincial Lotteries, U.S. lotteries have generated over $98 billion in revenue [1]. The fact that billions of dollars are continuously injected into a centralized and nearly obfuscated chance system is what inspired the team of Hyper Squid Game to develop the first fully decentralized and permissionless lottery pool jackpot game system using cryptographic blockchain technology integrated with modern pop-culture references. Every aspect of the Hyper Squid Game project is met with prioritization on technical transparency, innovative GameFi integration, and a robust infrastructure foundation for continuous scalability.

2. How The Game Works

The Hyper Squid Game smart contract only allows 456 players, or unique wallet addresses, to participate in each game. In order to participate, each player must deposit a minimum of 10 Arbitrum-native USDC tokens into the smart contract, or "pool", which grants the wallet address a single entry. Additional entries require a 10 USDC deposit per entry with no limit on the total number of entries a single

wallet address can have. The game winner is chosen based on a randomization sequence calculated from an array list stored within the smart contract on-chain containing the total number of entries for each wallet address. The winning player receives 90% of the entire pool with 10% distributed to the Hyper Squid Game team for expenses to further advance the project and buying back $HSG tokens and burning them as a deflationary measure. Games start and finish with a permissionless function built into the smart contract which checks if the hard-coded conditions are met to draw a winner and can be executed by any wallet address and scheduled for automated execution.

```
    /*
     * Function to draw the winner of the current game.
     * The function can be called by anyone, to ensure that
        the funds cannot be held hostage.
     * The random winning is derived from the block data of
        the current block.
     * 90% of the prize pool is paid to the winner, 10% is
       taken as a fee.
     * After the winnings have been paid out, the function
       prepares the contract for the next game.
     */
    function drawWinner() external returns (address) {
        require(entryCount > 0, "No entries purchased");
        require(block.timestamp >= lastDrawTime +
           COOLDOWN_PERIOD, "Cooldown period not elapsed");

        // Random number is chosen that is in the range of
           the entries bought for the current game.
        uint256 randomIndex = uint256(keccak256(abi.
           encodePacked(block.timestamp, block.number))) %
           entryCount;

        // Variable that keeps track of the amount of
           entires that have been processed already.
        uint256 processedEntryCount = 0;

        // Iterating over the list of the participants of
           the current game
        for (uint256 i = 0; i < participantList.length; i
           ++) {
            // Retrieving the participant wallet address
               from the list
```

```
21              address participant = participantList[i];
22
23              // Adding the amount of entires the participant
                    bought to the processsed entry count.
24              processedEntryCount += participantDataMapping[
                    participant].entries;
25
26              // Check if the current processed entry count
                    is higher than the randomly chosen number.
27              // If that is the case, it means that the
                    iteration has found the winner, so the
                    processes of paying out winnings begins.
28
29              if (randomIndex < processedEntryCount) {
30                  // Prize funds calculation (90% of the
                        prize pool)
31                  uint256 prize = (usdcToken.balanceOf(
                        address(this)) * 90) / 100;
32                  // Fees calculation (10% of the prize pool)
33                  uint256 tax = (usdcToken.balanceOf(address(
                        this)) * 10) / 100;
34
35                  // Prize funds are transferred to the
                        winner.
36                  require(usdcToken.transfer(participant,
                        prize), "Winner transfer failed");
37                  // Fees are transferred to the owner.
38                  require(usdcToken.transfer(owner, tax), "
                        Tax transfer failed");
39
40                  // Event emitted to keep track of wins.
41                  emit WinnerDrawn(participant, prize,
                        currentGameId);
42
43                  // Code to reset/prepare the contract state
                        for the next game.
44
45                  // Participant list is cleared.
46                  delete participantList;
47
48                  // Unique participant count is reset to 0.
49                  uniqueParticipants = 0;
```

```
                // Purchased entry count is reset to 0.
                entryCount = 0;

                // Last draw time is set to the current
                   block time, to ensure that the next draw
                    happens in an hour.
                lastDrawTime = block.timestamp;
                // The current game id is incremented by
                   one, to invalidate all of the data
                   related to the last game.
                currentGameId++;

                // Function returns the winner of the
                   raffle.
                return participant;
            }
        }

        // In case the function fails to pick a winner, the
            function is reverted.
        revert("Winner selection failed");
    }
```

3. Safeguard Security Measures

Players that deposit USDC into the smart contract can remove their entries at any time for a 10% early withdrawal penalty fee on the amount withdrawn using the implemented withdrawRaffle() function. This serves as both a trust mechanism and an emergency safeguard in the rare event a game experiences difficulties being completed.

```
    function withdrawRaffle() external {
        // Participant data is taken from participant data
           mapping
        Participant storage participant =
           participantDataMapping[msg.sender];
        // Variable to store the amount of entries the
           particpant has bought
        uint256 participantEntries = participant.entries;

        // Check to ensure that the particpant has bought
```

```
            entries in the current game
8           require( participant.gameId == currentGameId, "You
               have not participated in the current game");
9
10          // Check to ensure that the particpant currently
               has at least 1 entry in the game
11          require(participantEntries > 0, "You have not
               bought any entries");
12
13          // Calls to withdraw funds, 90% is sent back to the
                participant, 10% is taken as a fee
14          require(usdcToken.transfer(msg.sender,(participant.
               entries * ENTRY_PRICE * 90) / 100), "Withdraw
               transfer failed");
15          require(usdcToken.transfer(owner,(participant.
               entries * ENTRY_PRICE * 10) / 100), "Tax
               transfer failed");
16
17          // Global entry count is reduced by the amount of
               entires the partipant had bought
18          entryCount -= participantEntries;
19          // Participant entry count is set to 0
20          participant.entries = 0;
21
22          // Unique participant count is decreased
23          uniqueParticipants--;
24
25          // Event emitted to keep track of withdraws
26          emit EntriesWithdrawn(msg.sender,
               participantEntries);
27      }
```

4. HyperLiquid Integration

After the Hyper Squid Game token $HSG completes its auction, it will be used exclusively for pool entries instead of USDC with a portion of the 10% fees collected from each game continuing to be used to buy back and burn $HSG tokens for deflationary utility. The current smart contract deployed on Arbitrum will also be migrated to the HyperLiquid layer 1 blockchain once it is officially released and fully tested.

5. Project Roadmap

The future for Hyper Squid Game will continue to align with the functionality readily available at launch and detailed in the whitepaper, following the premise of precision execution for ideal user interaction. The plans for our roadmap are subject to change at any time, but remain heavily influenced by the community and the project's core principles.

5.1 Phase 1: Launch Hyper Squid Game and Token (Q4 2024)

The initial release of the fully decentralized and permissionless Hyper Squid Game lottery pool jackpot and corresponding $HSG token on HypurrFun is set to launch late Q4 - 2024. Readily available for release will be the first smart contract iteration of the game and an easy-to-use website interface to interact with.

5.2 Phase 2: Development Expansion (Q1 2025 and Beyond)

Further development to improve the game's user experience and attract new players will be a core focus for the project. The first priority will be a fully automated integration within the Hyper Squid Game X account to post when new games launch and end, which will include pool size and the winner's wallet address along with their number of entries and other game statistics. Following this closely will be the development of new games with innovative functionality and reward systems to greatly increase participation incentive and mass adoption. One of the many ways we plan to execute this is by rewarding winners of each game with an NFT version of their customized game character which is already stored on-chain on a per-game basis.

```
    /*
     * @param _suit The suit style number
     * @param _hat The hat style number
     * @param _number The player number
     * @param _eyes The eyes style number
     * @param _mouth The mouth style number
     * @param _shoes The shoes style number
     */
    function setAvatar(
        uint8 _suit,
        uint8 _hat,
        uint8 _number,
        uint8 _eyes,
        uint8 _mouth,
        uint8 _shoes
```

```
    ) external {
        // Retrieve participant data
        Participant storage participant =
            participantDataMapping[msg.sender];

        // Set all avatar customization values
        participant.suit = _suit;
        participant.hat = _hat;
        participant.number = _number;
        participant.eyes = _eyes;
        participant.mouth = _mouth;
        participant.shoes = _shoes;
    }
```

6. Conclusion

This whitepaper represents an in-depth analysis on the importance and technicality for a fully decentralized lottery infrastructure in which the Hyper Squid Game project has developed with permissionless and trustless execution. Through the time spent researching and developing Hyper Squid Game, it has become even more apparent that the native elements contained within the project fill a massive demand in a market sector with practically no other replication of similar principles.

References

[1] Boggs, J. (2023). How much money do lotteries generate and how much goes to winners, government. https://www.denver7.com/news/national/how-much-money-do-lotteries-generate-and-how-much-goes-to-winners-government. Accessed: 2024-12-29.

[2] Game, H. S. (2024). Hyper squid game website. https://hypersquidgame.com. Accessed: 2024-12-29.

[3] Wikipedia (2021). Squid game. https://en.wikipedia.org/wiki/Squid$_Game.Accessed$: $2024 - 12 - 29$.