FunFair Technology Roadmap and Discussion

June 6, 2017, v0.97

Jeremy Longley and Oliver Hopton

Contents

1 Introduction

The FunFair Technology Platform is an ambitious project designed to deliver next generation gaming anywhere in the world.This document discusses design goals, implementations and the upcoming roadmap.

For a full overview of the FunFair Platform, refer to our whitepaper at https://funfair.io/.

2 Disclaimer and Notice

Portions of the technology development discussed here may be protected by patents pending in the US, European Union, Asia or other markets worldwide. All information here that is forward looking is speculative in nature and may change in response to numerous outside forces, including technology, regulatory and market moves.

3 Design Goals

Our core mission is to build a platform and protocol for the next generation of responsive online gaming entertainment. All games on the platform must meet our strict criteria:

Provably Fair They should rely on the blockchain to verify gameplay.

Bankless They should rely on the blockchain to store transactions.

Fast There shouild be zero latency between the user pressing any UI button and seeing the result.

Flexible The games should support any type of player decision during a game session, including:

- Varying the size of bets from game to game
- Playing immediately with winnings
- Adding funds mid-session

3.1 Funness

We don't want just simple, boring "pick a number" games. These are endemic to blockchain gaming, and they are just not fun. We are fun. The gaming industry has spent hundreds of years refining fun games for their customers – we think it's wrong to ignore those lessons. Therefore, the platform must

- Support multi-state games like Blackjack, with the ability to split, double and play multiple seats
- Support games where state persists between individual rounds like Craps

4 Random Number Generation

At the core of any casino game is a random number generator (RNG). All software-based RNGs are effectively pseudo-random number generators (PRNGs).

PRNGs on blockchains have many problems.

Per Transaction Some blockchain games generate randomness one play at a time, a slow and costly approach that isn't viable for players or operators. Additionally, these typically require a daily "commit and reveal" of secret information. The gamer can verify after-the-fact that the operator did not cheat, but must trust during the game. Old-style games that use this setup like SatoshiDice may cost up to $1-5 to play before any bet is staked and take minutes to hours per bet depending on block time how full the blocks might be.

Oracles Other blockchain-based online games use oracles to generate randomness. These are also slow, limited to blockchain confirmation speeds, often requiring multiple blocks to be mined before results are available, and add difficult incentivization questions about the operators, their employees and servers.

Block Hashes The worst use block hashes for random number creation; slow and fundamentally vulnerable to attack by miners, by their very design.

None of these technologies can create compelling games in 2017.

5 Our Solution: Fate Channels

We have invented a provably fair system using state channels containing pre-committed partial RNG seeds provided separately by the player and the operator. These partial seeds are comitted to the blockchain at the start of the gaming session.

State channels work by allowing participants to engage in a rapid back-and-forth countersigning of updated "claims" on an escrowed amount of funds. The Bitcoin Lightning protocol was the first to popularize the idea. On EthereumVM-capable blockchains, State channels are very simple to create and implement – a smart contract holds escrowed funds, and then releases when the state channel participants request it.

During gaming, we create instead a "Fate Channel"; a State channel with the added ability to verify a progressive reveal scheme by both parties, advancing a deterministic ("fated") but unpredictable sequence of random numbers.

Details of our Fate Channel implementation will follow our token sale; for now we feel it gives us and our token holders a competitive advantage to keep implementation details private a little bit longer.

6 Development History

We started with a relatively complex Slot game: 30 win-lines, wild cards, scatter symbols and a bonus mode featuring free spins.

Many Slot implementations online determine the position of the reels by first deciding on an outcome, then animating the reels to an appropriate position, but this approach restricts the complexity of future games. It's also not the way slot machines work in real life, so this might be considered a 'fake slot', more akin to a Dice game with slot graphics overlaid on top.

A 'pick a number then animate' solution is a technology dead end. More complex games like Blackjack and Roulette do not easily translate to this method, so we implemented it the 'hard way' in preparation.

6.1 FunFair Slot v0.1

We encoded the rules of the Slot machine on the blockchain. Initially, we built a system in which a smart contract picks a future block to be used as the seed for a random number generator (RNG). When the appropriate block is mined, the contract uses the hash of the previously mined block, salted appropriately to generate entropy and then walks through a state machine, encoded into the smart contract. This state machine determines the position of each of the reels of the Slot machine.

As discussed, this is a bad method to get random numbers. This method has risks for operators if miners are gaming – miners can have incentive to throw away a mined block if it contains a loss, depending on a number of extrinsic and intrinsic factors.

6.2 Slot v0.1 Enhancements

We then extended the basic play in two ways:

Multi-Spin The player can "insert" funds for more than one spin – the entropy generated can be used to determine a sequence of spins rather than just one at a time. This is the core of our first innovation that yields instant gaming.

Free Spins Once we were able to create multiple spins through the initial seed, we extended the capabilities of the machine to add spins, still seeded off the same random number.

This system was built and works as intended. We built a stylish web-based front end, including symbol animations and effects to give an indication of the final quality of product we can build. See pictures and links to videos at https://funfair.io/whitepaper.

6.3 Problems with v0.1

The initial project was a success and we proved that it was possible to build a high quality, fun, instantly responsive Slot machine on the blockchain.

This system failed our core requirements. It was exploitable by miners, did not allow mid-game decisions and was gas cost prohibitive.

Although we did come up with a set of enhancements that prevented miner cheating (by blinding knowledge of the RNG at the moment of mining with a commit/reveal scheme but this method was abandoned in favour of the superior solution.)

6.4 Design Goals for v0.2

The PRNG scheme in v0.1 did not support our goal of provably fun, fast, fair and flexible games. Looking for an alternate solution, we incorporated three existing areas of research from the blockchain and gaming communities.

State Channels State Channels allow micro-transactions to be handled between parties directly, off-chain; funds are committed and locked in a smart contract in advance. Then transactions occur between the parties, progressively signed by all. The final state is committed back to the chain in a way that proves all parties agree on the outcome.

Provably Fair Here the basic concept is that some entropy is committed to the blockchain in advance, but encrypted. Once the gameplay has occurred, this entropy can be revealed to have been the source of the RNG used for gaming - which, in combination with encoding the rules of the game in a smart contract, can be used to prove the fairness of the game

Reverse Hash Chains To create a hash chain, choose a secure initial seed, hash it, hash the hash, repeat many times and reveal the last hash. At the time of reveal you have created a series of secret random numbers that can be revealed sequentially, one at a time and backwards. The revealer can't cheat or change their mind without breaking the hash algorithm – they must reveal the prior number in sequence, and this number can be verified easily as the correct 'prior' secret number.

6.5 Fate Channels v0.2

To power the random number generation behind FunFair's Slot machine – and ultimately behind all games on our platform– we invented a random number generator that operates off-blockchain. It exists as a state channel for the duration of the gaming session and supports realtime messaging between the FunFair client and server. We call these channels "Fate Channels."

6.5.1 The Channel Setup

Fate Channels support communication throughout a gaming session between the Player (client) and the Casino (operator). They Provide a fast low-cost method for random number generation, starting gaming sessions, ending them and settling with smart contracts on the blockchain.

For each game, a smart contract API call is issued encapsulating the rules of the game. The overarching Fate Channels contract primarily does two things: starts a gaming session, and settles, (or ends) one.

6.5.2 The Client

The Client is a web-based, Javascript and HTML5/WebGL application that can communicate with the blockchain and the Server. In the future clients could be native applications on mobile, but for now they can work directly inside a mobile browser, giving access to the widest possible range of customers.

6.5.3 The Server

The Server is very similar to the Client but headless, with no user interaction.

After the Player initiates a gaming session, both the Client and Server create reverse hash chains. The Fate Channel contract verifies all interactions from both players and creates a session on the blockchain, locking funds and posting an `Event` to the blockchain.

6.5.4 The Protocol

As gaming progresses, the Client and Server trade messages, signed by the sender.

If the state of the session has advanced in a given message, the new state is signed and the previous state (already signed by the other party) is co-signed. For each outcome in a gaming session, the Server creates the RNG by pulling the next hash from its Reverse Hash Chain, combining it with the next client hash (provided by the client), and taking the hash of that.

It then runs the game logic using this RNG – determining an action in the gaming state machine. Meanwhile, the Client uses the same hashes to generate the RNG in the same way the Server did. It uses its own implementation of the game logic to verify winnings and other game outcomes match those of the Server.

To end a session, the Player presses a cash out button. The client signs the most recently traded state, the server co-signs and calls the Fate Channel contract with an "End Session" message. The contract extensively verifies all relevant data. If all is good, the contract pays out the final balance and the session is ended.

6.5.5 On-Chain Verification

This initial implementation allows for secure, deterministic, fair gameplay. However, we would also like to validate this on-chain. In order to do this, we need two additional contract methods:

Chain Verification To verify each of the the Reverse Hash Chains created by the Client and Server, the last hash used is posted to the chain - the Fate Channel can then hash this the correct number of times and recreate the committed 'final hash' of the chain.

Game Verification To verify an individual game, we implement the game state machine in a separate contract on the blockchain, one per game type. This can be a constant function, taking the initial seeds, the game and the game output, and returning `true` or `false`. Because constant functions do not modify the blockchain, they are free to call and execute for all participants.

7 Fate Channels v2

The next round of development will make two improvements.

Ephemeral Channel Address In Fate Channels v1, the Client generates an Epheremeral key to sign channel messages. Clients may forget their address signing key if they wait too long or don't wish to re-sign a transaction from their web3 client to claim funds. Clients can leave because interface requests add friction or just because they become distracted and close a browser window. The client signing key could be stored encrypted in the Fate Channel contract alongside the session. We believe this could be extended to recover partial sessions as well.

Unified State Machine Fate Channels v1 store logic in three places: the smart contract, the Client and the Server, and at times these are implemented in different languages – Solidity, Javascript and (currently) C#. We are exploring how to unify these. Possibilities include a standardized state machine structure or instrumenting Client and Server to use only Ethereum blockchain constant functions.

8 Fate Channels v3: Multi-State Games

The FunFair Slot machine is a proof that Fate Channels can power a faster, fairer game that doesn't rack up transaction costs or spam the blockchain. We are the only experienced gaming company in the world to get this far. We will go further. Slot machines, dice games and Roulette are "fire and forget": Once a player spins the wheel or rolls the dice, there is nothing they can do mid-play to change their bet or affect the outcome.

"Multi-state" games are more popular globally. When players can affect the game in some way, perhaps by changing their wager (Blackjack), or basing a new wager on results of a prior one (Craps), they are more engaged.

To support multi-state games like Blackjack, FunFair will extend the Fate Channel protocol. Fate Channels v3, incorporating these changes are set to release Q3 2017.

9 Fate Channels v4: Fully Peer-to-Peer

Fate Channels imagine a Server and a Client. FunFair intends to allow operators to run a Server directly as part of the simple two click process. This ability to allow anyone to launch and operate their own gaming environment is a key part of our value proposition: making it easy for long tail operators, private groups, celebrities and others to operate quality games is what we do.

We would like to remove the Server component completely from the Fate Channel specification, and believe we have a way to do that. If we are successful, then these operators will get full benefits from the distributed and decentralized nature of the blockchain.

This is a technically challenging innovation, and we're only 95% sure we can do it, but early indications are good, and the benefit to all FunFair stakeholders is immense.

We are currently slating Q4 2017 for full P2P play with no servers at all – all while meeting our key requirements that each game be fun, fast and fair.

10 A Note on Threshold Cryptography

Threshold Cryptography - for instance (BLS), popularized by String Labs for their Dfinity project, holds out the promise of a truly secure random number generated by a group of participants, at least one per block. These cryptographic primitives are supported in Ethereum's Metropolis release (and also add support for zk-Snarks) and other privacy primatives.

We may update our mechanisms to incorporate this technology, but we will still rely on Fate Channels for the "instant" part of the gaming – gas costs will be prohibitive for using these RNGs more often than at the beginning and end of channel creation.

We are excited by the pace of development of the Ethereum infrastructure and are continually evaluating new features and research that could enhance our offering further.

11 More Information

For more technical information, check our website at https://funfair.io or email us on info@funfair.io