SoftLink

SoftLink Protocol
softlinkprotocol@protonmail.com
www.softlink.finance

Abstract

In this paper, we introduce SoftLink, a software protocol that creates a platform to link together a network of liquidity providers and searchers to benefit from successful MEV strategies executed on an EVM-compatible network via smart liquidity management of reserves.

1. Introduction

Background

Decentralized Finance (DeFi) introduced novel concepts that would alter the perception of the bounds of finance forever.

Core concepts such as liquidity pools and liquidity mining gave birth to an industry where anyone can benefit from interacting with these new smart reserves backed by a pair of coins and/or tokens the creator saw fit.

Additionally, with the introduction of Flash Loans, and Flash Swaps, from protocols such as Marble, AAVE, and UniSwap, any participant on the Ethereum network gained the ability to borrow funds, perform a variable amount of actions with the funds, and return the loan plus a fee all in one transaction.

It’s clear to see how such an innovation would increase the opportunity for MEV searchers to seek and produce more profitable strategies.

But what is MEV?

Who/what are MEV searchers?

To understand MEV, we must briefly review its historical origins.

Starting from a purely peer-to-peer version of electronic cash, to a “decentralized computer”, to a movement to build an open, global, and interconnected financial system, the ecosystem fueled by innovations in distributed computing and blockchain technology changed the way humans interact with financial systems forever.

While the story of Decentralized Finance spans before DeFi Summer, in this paper we’ll touch on the impact of Marble, AAVE, and UniSwap, given their influence on the creation of the SoftLink protocol.

Marble

The Marble Protocol is described as the following: “An open-source bank on the Ethereum blockchain. The Marble smart contract owns Ether and tokens and makes its funds available to provably fair, low-risk protocols that require a lender.”

Marble was the first protocol built on top of the Ethereum blockchain that introduced “flash lending” which allowed for anyone to borrow ETH and/or ERC20 tokens for use cases such as DEX (Decentralized Exchange) arbitrage opportunities.

Sadly, if one reviews their GitHub repository for flash lending, efforts concluded back in 2018; however, the team moved onto working on Humanity “a Decentralized Autonomous Organization (DAO) that governs a registry of unique humans on Ethereum” so they’re still very active but concluded their efforts on expanding decentralized, financial lending technology any further.

AAVE

After two years, in January 2020, the AAVE protocol introduced flash loans on their open source and non-custodial liquidity protocol.

In fact, the founder Stani Kulechov gave a lecture on flash loans at the CryptoCompare summit during March of that same year without realizing how much impact their flash loan technology would have later that summer.

With DeFi Summer in full effect, protocols such as UniSwap and AAVE, saw an insane rise in protocol usage from wallets on the network to the point where AAVE was issuing more than $100 million in flash loans each day.

Not too much longer in the future, AAVE would process a flash loan of about $200 million in one transaction.

Uniswap

Uniswap, a very popular decentralized exchange, introduced their implementation of flash loans/swaps in 2020.

Given recent data found on Dune Analytics, Uniswap V2 Cumulative Flash Swap Volume has **reached over $6 Billion USD with Daily Flash Volume reaching over $600 million in December 2020**.

MEV (Miner/Maximal Extractable Value)

Given these tools, important financial concepts such as MEV can be applied to the Ethereum network.

Described at MEV day in Amsterdam by Phil Daian, one of the co-authors of “Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges”, MEV “refers to the total amount of Ether miners can extract from manipulation of transactions within a given timeframe, which may include multiple blocks’ worth of transactions.”

MEV searchers will be operationally defined as any entity who executes strategies for the sole purpose of employing any form of MEV technique such as front-running, back-running, sandwiching, liquidity sniping, long-tailing, etc.

Miners must be considered prior to specific types of MEV searchers who are not miners because miners in turn can be MEV searchers.

To describe it simply, miners create/mine new blocks by packing a certain amount of valid transactions into one.

Since they have the ability to receive and pack transactions into a block, they have the power to order the transactions in a block however they see fit; therefore, since one can assume that a miner's core incentive is to extract the most value out of the network, most, not all, miners include/organize transactions such that they optimize for the maximum amount of value that can be generated for mining the self-alternate block(s).

Searchers, who are not necessarily miners, can express transaction ordering preferences in order to capture some value at a higher level and not just at a low level (mining new blocks).

For example, a searcher can connect and listen to the same mempool that miners, from a mining pool, pull pending transactions from in order to peek into certain actions from a retail trader interacting with a decentralized exchange such as Uniswap.

They can deploy smart contracts to execute a strategy such as sending out transactions to sandwich a particular trade to the mempool with a specified amount of gas and/or an attractive tip for the miner such that the miner includes their transactions in the block they mine/broadcast to the network.

That searcher can then leverage protocols who offer flash loans to use temporary debt in the life span of the transaction to perform a swap on UniSwap of greater value and pay for costs such as the total gas consumed, the miner tip, the overall loan fee, etc.

Keep in mind, that is only one example of one type of strategy.

Purpose

Even though existing protocols provide solutions for flash loans and liquidity provisioning and management, they lack critical features that allow for MEV searchers to reduce the cost of executing profitable strategies.

Profitable strategies do not have to benefit MEV searchers only, and solutions do not have to provide a service that minimizes MEV profits in order to account for the side effects of running profitable strategies.

SoftLink addresses this problem by providing a solution that benefits both liquidity providers and searchers through a smart liquidity reserve provisioning and management system that searchers can tap into to run MEV strategies leveraging flash loan technology.

In the following sections we dive deeper into both the liquidity reserve system and our flash loan technology solution that reduces the cost for MEV searchers.

2. Reserves

Mechanism Design

Reserves are pretty straightforward.

In order for a reserve to exist one must create a reserve via the CoreReserveFactory:

```
struct CreateReserveParams {
    CoreReserve.ReserveType reserveType;
    bool isNativeTokenReserve;
    address reserveToken;
    uint256 reserveTokenInitialAmount;
}
```

```
function createReserve(CreateReserveParams calldata params) returns (address)
```

Once a reserve is created, one can provide liquidity to the selected reserve via a deposit:

```
function deposit(uint256 _depositAmount, address _creationDepositor) returns (bool)
```

Whenever one is ready to withdraw liquidity, they can simply do so via a withdrawal:

```
function withdraw(uint256 _withdrawAmount) returns (bool)
```

For advanced/technical liquidity providers, the first stream of income introduced is the ability to instruct the protocol to provision your liquidity to earn additionally yield through the initial list of protocols we trust with one's deposited liquidity:

1. AAVE: https://aave.com
2. Compound: https://compound.finance
3. Euler Finance: https://www.euler.finance
4. Iron Bank: https://app.ib.xyz

SoftLink will deposit and manage 50% of the given deposit amount into the chosen protocol (the amount is subject to change in the future).

Given the flexibility of ERC-20 tokens, the protocol accounts for both LP tokens and Non-LP Tokens such as ETH/DAI and ETH (reserve creators only need to specify the address of the specific token).

Additional 3rd-party protocols can be appended to the list in the future through the governance structure and process outlined later in the paper.

3. Flash Loans

API Design

Each deployed reserve contains the following two accessible functions for performing flash loans:

```
struct FlashLoanParams {
    address receiver;
    uint256 amount;
    bytes params;
}
```

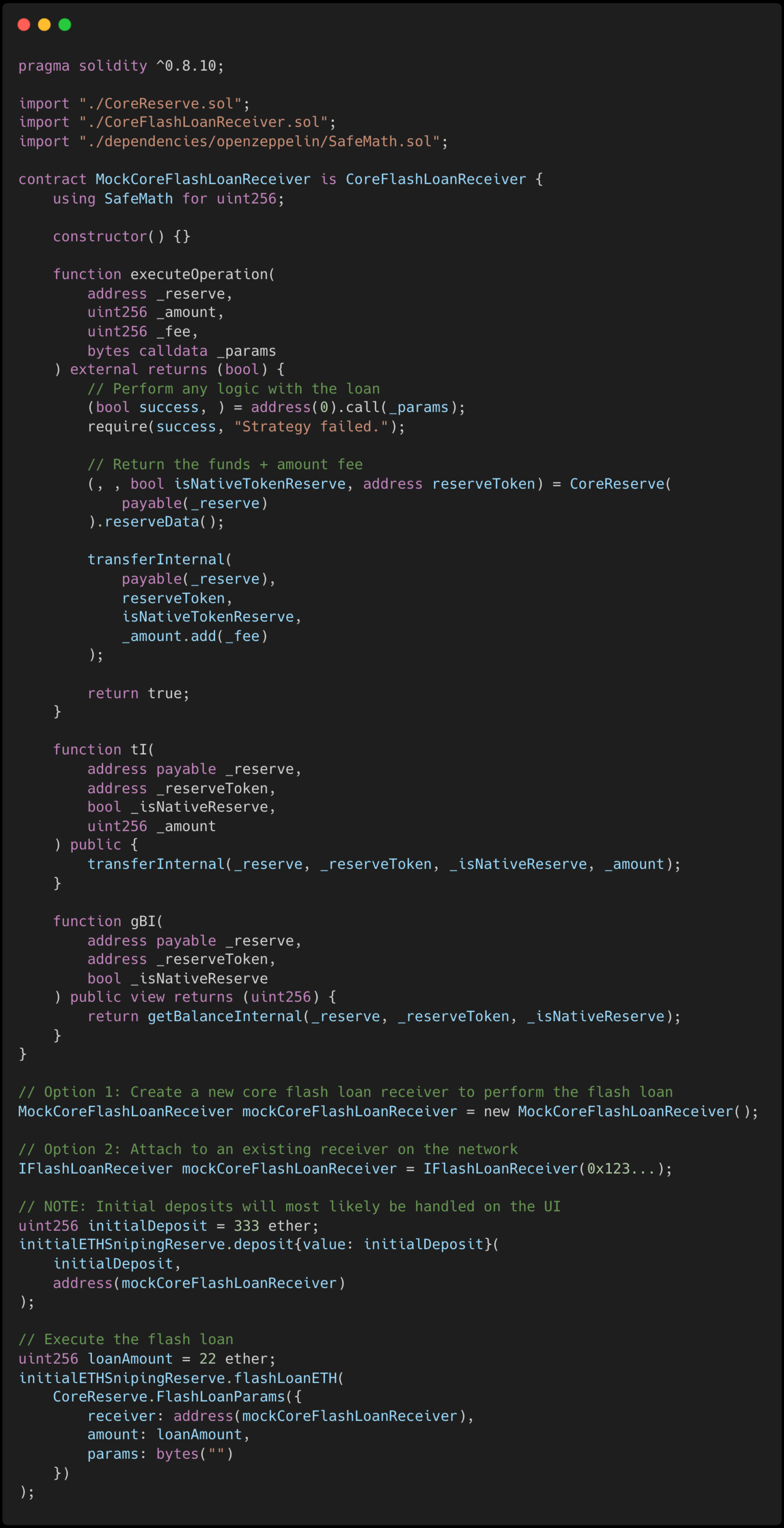
```
function flashLoanETH(FlashLoanParams calldata _params) public returns (bool)
```

***Can only be executed on reserves configured to manage currencies native to the network (ETH, FTM, NEAR, SOL, etc.)**

```
function flashLoanERC20(FlashLoanParams calldata _params) public returns (bool)
```

***Can only be executed on reserves configured to manage ERC20 tokens (DAI, AAVE, COMP, ETH/DAI, etc.)**

The following is an example of how one can design a simple smart contract to perform a flash loan on the protocol:



```
pragma solidity ^0.8.10;

import "./CoreReserve.sol";
import "./CoreFlashLoanReceiver.sol";
import "./dependencies/openzeppelin/SafeMath.sol";

contract MockCoreFlashLoanReceiver is CoreFlashLoanReceiver {
    using SafeMath for uint256;

    constructor() {}

    function executeOperation(
        address _reserve,
        uint256 _amount,
        uint256 _fee,
        bytes calldata _params
    ) external returns (bool) {
        // Perform any logic with the loan
        (bool success, ) = address(0).call(_params);
        require(success, "Strategy failed.");

        // Return the funds + amount fee
        (, , bool isNativeTokenReserve, address reserveToken) = CoreReserve(
            payable(_reserve)
        ).reserveData();

        transferInternal(
            payable(_reserve),
            reserveToken,
            isNativeTokenReserve,
            _amount.add(_fee)
        );

        return true;
    }

    function tI(
        address payable _reserve,
        address _reserveToken,
        bool _isNativeReserve,
        uint256 _amount
    ) public {
        transferInternal(_reserve, _reserveToken, _isNativeReserve, _amount);
    }

    function gBI(
        address payable _reserve,
        address _reserveToken,
        bool _isNativeReserve
    ) public view returns (uint256) {
        return getBalanceInternal(_reserve, _reserveToken, _isNativeReserve);
    }
}

// Option 1: Create a new core flash loan receiver to perform the flash loan
MockCoreFlashLoanReceiver mockCoreFlashLoanReceiver = new MockCoreFlashLoanReceiver();

// Option 2: Attach to an existing receiver on the network
IFlashLoanReceiver mockCoreFlashLoanReceiver = IFlashLoanReceiver(0x123...);

// NOTE: Initial deposits will most likely be handled on the UI
uint256 initialDeposit = 333 ether;
initialETHSnipingReserve.deposit{value: initialDeposit}(
    initialDeposit,
    address(mockCoreFlashLoanReceiver)
);

// Execute the flash loan
uint256 loanAmount = 22 ether;
initialETHSnipingReserve.flashLoanETH(
    CoreReserve.FlashLoanParams({
        receiver: address(mockCoreFlashLoanReceiver),
        amount: loanAmount,
        params: bytes("")
    })
);
```

4. Gas-Free Flash Loans

Problem

Highly active and successful searchers are constantly executing complex strategies; however, the obvious issue with strategies involving more complexity is the gas cost associated with each strategy.

Additionally, strategies leveraging transactions that involve flash loans increase the total amount of gas required (given that flash loans must execute several operations in sequence successfully).

Solution

Inspired by the design of Executor Operators developed by engineers at Gelato Network, our protocol designed a fairly simple, but effective, system to offload the additional gas included in strategies leveraging flash loans through a proxy network.

Designed for high availability, scalability, and feasibility, the proxy network deploys stateless processes designed specifically to redirect requests to perform flash loans within the protocol's network of reserves to minimize strategy gas costs.

Leveraging the Gelato Relay SDK, the engineers at Gelato Network guarantee that flash loan transactions are mined fast, reliably, and securely.

Proxy API Design

In order for a searcher to interact with the proxy network, the following endpoint is publicly exposed and implements rate-limiting:

POST /api/v1/eth/flashloan/execute

Request Body Schema (JSON):

```
{
    receiver: String,
    amount: Number,
    params: String (Hex String - ethers.js),
    isNativeFlashLoan: Boolean
}
```

Response Body Schema (JSON):

```
{
    success: Boolean,
    gelatoData: {
        taskId: String (Hex String - Review Gelato Relay SDK notes)
        message: String
    }
}
```

Keep in mind that if the address given is not a valid CoreFlashLoanReceiver (such as in the example above), the execution will fail.

POST /api/v1/eth/flashloan/querytaskstatus

Request Body Schema (JSON):

```
{
    taskId: String (Hex String - Returned from /api/v1/eth/flashloan/execute call above)
}
```

Response Body Schema (JSON): https://docs.gelato.network/developer-products/gelato-relay-sdk/request-types?q=task+status#querying-task-status

TLS/SSL

Given that searchers care deeply about keeping strategies in the shadows to maximize total profitability, all requests are encrypted in transit via the widely known protocol HTTPS.

The importance of implementing HTTPS lies in its ability to mitigate man-in-the-middle attacks where sophisticated attackers can intercept network packets in transit from the sender (client) to the receiver (server).
HTTPS implements public key cryptography which allow for messages to be encrypted; therefore, if attackers are successful in assuming control of a remote device in between the sent packet's origin and destination, the message would be ultimately unreadable.

Transaction Gas Threshold

The protocol initially sets a transaction gas threshold, but this threshold is open to being updated if public proposals pass in governance.

The transaction gas upper bound is initially set at 0.21 GWEI (210,000,000) or 100x the minimum amount of gas an operation on Ethereum will use.

5. Cross chain functionality

The overall DeFi industry is moving towards Cross-chain functionality as of 2022; therefore, our protocol will be building solutions to enable cross-chain liquidity provisioning and management alongside the existing flash loan solutions.

As time progresses, and as the protocol grows, more and more information will be released regarding our system design for cross-chain functionality and our rollout strategy.

Enabling an experience for searchers to pay the minimum, to no gas, for performing profitable strategies is one of the protocol’s top priorities and many L2 solutions are paving the way for this experience to be possible.

Future partnerships will be established to aid the implementation of deploying SoftLink across multiple chains as well.

6. Fee Distribution

For every flash loan, and for certain critical protocol activities such as depositing and withdrawing from reserves, a fee is applied to the transaction

For flash loans, the loaner must provide the loan amount + a fee in order to execute a successful transaction.

For V0, both the base amount flash loan fee, and the base amount protocol transaction fee, is 0.22%

Furthermore, the base amount flash loan fee is divided so portions of the fees are distributed to the entities listed below:

Liquidity Providers

50% of amount fee

Core Fund Reserve (CFR) Donations

25% of amount fee

Protocol Revenue

20% of amount fee

***The remaining 5% of the amountFee will increase the total balance of the pool**

7. Core Fund Reserve Distribution

Purpose (Transparent Charity System)

The purpose of the Core Fund Reserve (CFR) is to allocate its earnings at the end of every CFR distribution period to a specific charity organization that's approved by the community.

The CFR distribution period is subject to change through governance but the initial period is set to 1 year (365 days in the Gregorian calendar).

The Core Fund Reserve is managed by the protocol admin's multi-signature wallet (Gnosis) that will only transfer funds to and from if specific governance proposals pass.

Those governance proposals are labeled TCS (Transparent Charity System) proposals where the proposal must outline the following:

1. The charity organization where the distribution period of CFR funds will be sent to.
2. Why that specific organization should receive the distribution period's funds.
3. A link to an official message from the charity organization dictating where the funds will be transferred to ensure full transparency for the community (the world should see how the Charity organization manages the received funds).

8. Governance

Governance will be managed by Snapshot, an efficient and well known decentralized voting system, and Tally, a platform to deploy and operate DAOs.

More specifically, Snapshot is “an off-chain gas-less multi-governance client with easy to verify and hard to contest results.” while Tally is “a DAO operations platform that helps people start, join, and grow decentralized organizations.”

To integrate with Snapshot, our protocol will need to add a record on ENS to allow votes to be viewable at the created ENS address and either a governance token and/or NFT for voters to be eligible to vote.

To integrate with Tally, we will follow the deployment guides within their docs on docs.tally.xyz.

Therefore, at first, governance will be handled by the protocol admin’s multi-signature wallet.

However, an official governance token with fleshed out tokenomics determined after careful financial consideration with the protocol’s stakeholders is suggested over the prior.

9. Looking Forward

Creating value from successful MEV strategies is only the first step for the multi step plan to onboard everyday users of the web.

More and more updates will be announced on our Medium blog regarding SoftLink’s future plans.

10. Conclusion

SoftLink is a protocol that enables smart management and provisioning of liquidity reserves as well as an application programming interface (API) for advanced flash loans, allowing both liquidity providers and searchers to profit from successful MEV (Miner/Maximal Extractable Value) strategies executed on an EVM-compatible network.

Existing protocols offer answers for managing liquidity and flash loans, but they are missing key components that would help MEV searchers execute profitable methods more cheaply.

In order to account for the negative effects of implementing successful strategies, solutions do not necessarily have to provide a service that lowers MEV earnings.

To solve this issue, SoftLink offers a system for the provisioning and management of intelligence liquidity reserves, which searchers can build MEV strategies on top of via flash loans and future fleshed out features.

This system benefits both sides of the network: liquidity providers and MEV searchers.

Additionally, our protocol created a very straightforward but efficient mechanism to unload the extra gas included in strategies employing flash loans through a proxy network.

The proxy network deploys stateless processes specifically designed to reroute requests to perform flash loans within the protocol's network of reserves to eliminate gas costs.

The proxy network is designed for high availability, scalability, feasibility, and transport security through HTTPS.

References

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” https://bitcoin.org, bitcoin.pdf, 2008.
[2] Blockchain at Berkeley, “Ethereum and Smart Contracts: Enabling a Decentralized Future,” https://docs.google.com/presentation/d/1Rv7uEO5NpQJmJByKAj35o-_PNDKucUDJZMJXVkr8yDA/edit, 2021
[3] Blockchain at Berkeley, “Defi / Xcelerator,” https://docs.google.com/presentation/d/1bp2wDWHmNja6DoCr4-CBytGYVI_aOI_wjzzLaiSiphw, 2021
[4] Finematics, “History of Defi - From Inception To 2021 And Beyond,” https://finematics.com/history-of-defi-explained, January 4th, 2021
[5] Gelato Network, “Executor Operators,” https://docs.gelato.network/introduction/executor-operators
[6] Gelato Network, “Gelato Relay SDK,” https://docs.gelato.network/developer-products/gelato-relay-sdk?q=task+status
[7] Ethers.js, “Hex Strings,” https://docs.ethers.io/v4/api-utils.html#hex-strings
[8] Surveillance Self-Defense, “A Deep Dive on End-to-End Encryption: How Do Public Key Encryption Systems Work?,” https://ssd.eff.org/en/module/deep-dive-end-end-encryption-how-do-public-key-encryption-systems-work, November 29, 2018
[9] Snapshot Labs, “README.md,” https://github.com/snapshot-labs/snapshot, June 10, 2022
[10] Matt Hussey, “What is Snapshot?,” https://decrypt.co/resources/what-is-snapshot-the-decentralized-voting-system, June 4th, 2021
[11] Max Wolff, “Introducing Marble,” https://medium.com/marbleorg/introducing-marble-a-smart-contract-bank-c9c438a12890, July 16, 2018
[12] richmcateer, “Humanity,” https://github.com/marbleprotocol/humanity, July 7th, 2019
[13] Phil Daian, “mev wat do next”, https://docs.google.com/presentation/d/1mUID0JDCiJz1_U4Jn05A0gleliZu7UZQOR86OUYFW3g/edit#slide=id.g12545bcc526_0_368, April 22nd, 2022
[14] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, Ari Juels, “Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges”, https://arxiv.org/pdf/1904.05234.pdf, April 10th, 2019
[15] Hasu, “#29: Interview with a Searcher – with MEV Senpai and Hasu + transcript,“ https://uncommoncore.co/29-interview-with-a-searcher-with-mev-senpai-and-hasu, July 19th 2021
[16] “Get Started” https://docs.tally.xyz/ August 4, 2022

Revision History

July 3rd, 2022
July 5th, 2022
July 6th, 2022
July 7th, 2022
July 14th, 2022
July 16th, 2022
July 26th, 2022
July 28th, 2022
July 29th, 2022
August 1st, 2022
August 5th, 2022
August 7th, 2022
August 18th, 2022
August 26th, 2022
August 29th, 2022
August 30th, 2022
September 1st, 2022
September 3rd, 2022
September 20th, 2022
September 30th, 2022
November 20th, 2022