KeeperDAO

The self-iterating liquidity underwriting machine.

Ethereum is a dark forest. It is a place full of advanced predators waiting to eat every bit of profit that can be found. Whether it is an arbitrage opportunity, a liquidation, or something else, if it makes profit, you can be sure the predators are watching.

You are clever and conniving. You have found an opportunity to make profit on-chain, an arbitrage between two DEXs, just waiting to be taken. You prepare a transaction and submit it. But before your transaction can be confirmed, it must first wade through the mempool, an arena where pending transactions fight for a place in the next block. Here, your transaction is visible to everyone. Anyone can simulate it. Anyone can analyse it. Anyone can copy it.

You are clever and conniving. You obfuscate all your transactions. You bounce them through multiple contracts, callable only by you. You encode their inputs. You even break them down into multiple disjoint steps. But Ethereum is a dark forest, and your transaction is the prey. There are advanced predators within the forest, with advanced weapons, able to trace every transaction, every internal contract call, and every instruction. The predators can discover which parts of any transaction are extracting profit, and they can copy those exact parts.

You are clever and conniving. You know how to fight the gas war. When the predators find your transaction, and submit their own modified versions, you bid, you counter-bid, you randomly raise, you feign, and you grim-trigger. You eat into your own profits to raise the gas price ever higher. Sometimes you win, sometimes you lose. But, no matter the result, the miner *always* wins. It wins because it claims the gas. The gas that you must spend to within an inch of profitability. The gas that brings you a loss when you fail to cancel your transactions in time. You take all the risk, and the miner takes all the profit. So we call these opportunities MEV: *miner extractable value*.

You are clever and conniving. But you are bound to the laws of the blockchain. The miners *are* the blockchain. Long after your transaction is confirmed, they can reorganise blocks, reorder transactions, or drop them entirely. The miners have the power to undo what has been done, and take profit wherever it can be found. We have convinced ourselves that we are predators, and the miners are feeding off the gas that we burn. In reality, we are scavengers, fighting over what is left to us by their good graces. If Ethereum is a dark forest, then the miners are the trees, and the forest itself is the most advanced predator of all. It is time to burn the forest down.

Abstract

KeeperDAO is a game-theory protocol designed to capture the maximum possible on-chain profit, whether it is arbitrage, liquidation, or something else. This is achieved in three ways:

The *Hiding Game* is introduced to allow users to route their liquidity through specialised proxies, whether it is for trading or lending. These proxies automatically extract any profit that can be extracted from arbitrage or liquidation and return those profits back to the user. The result is minimum slippage on trading, and minimal losses on liquidations.

The *Coordination Game* is introduced to incentivise arbitrageurs and liquidators, collectively known as keepers, to collaborate with each other, instead of fighting gas wars. By playing the game – taking turns, redistributing profit, and discouraging bad actors with tit-for-tat strategies – keepers can earn more profit than if they were to act as an individual.

The *Incentive Game* is introduced to incentivise specific behaviours that are critical to the efficient and effective functioning of KeeperDAO. This includes providing liquidity, bootstrapping into the Coordination Game, playing the Hiding Game, and participating in governance.

Background

Miner Extractable Value

MEV – an abbreviation of miner extractable value – is a general term for profit that can be extracted by miners. This extraction can happen when miners reorganise blocks, reorder transactions, front-run, and tailgate. For example, a user that submits an order to Uniswap will often find their transaction sandwiched between two other transactions; forcing the maximum amount of slippage, and extracting the maximum amount of arbitrage.

This is clearly undesirable for users. But it does not end there. In their paper *"Flash Boys 2.0"*, Phil Daian et al. highlight the risks of MEV to not only the user, but to the consensus layer itself. The reorganisation of blocks, the reordering of transactions, and – even momentarily – the censorship of transactions can have profound impacts on the blockchain, and breaks the assumptions of many applications.

The existence of generalised MEV solvers has also been hypothesised. These solvers are capable of analysing transactions, even those that have been obfuscated, to break them apart and reassemble them such that all profit that can be extracted goes to someone else. This capability has given rise to the idea of *the dark forest*; an environment where detection by one of these MEV solvers means having your profit devoured. In his blog post *"Ethereum is a Dark Forest"*, Dan Robinson walks the reader through an encounter with one of these MEV solvers, realising their existence today. In his attempt to recover some lost funds, his transaction was discovered, and an automated bot altered his transaction and submitted it for itself. Never before had the contract in question been called in this way – by anyone – despite the bot having a long history of transactions. It was clearly the result of a generalised MEV solver being put to work.

Priority Gas Auctions

While we refer to these kinds of profit opportunities as MEV, it is not always done by the miners (although there are certainly opportunities that can only be taken by miners, thanks to the nature of block production). More commonly, MEV ends up being extracted by arbitrageurs and liquidators (also known as *keepers*). Curiously, though, the result is the same: miners extract the majority of the available profit. To understand why this happens, it is important to understand how PGAs – or *priority gas auctions* – affect transaction ordering and profitability.

When a keeper discovers a price imbalance between two DEXs, or an unhealthy position on a lending protocol, it is rarely the case that no one else has noticed it. This leads to a competition where keepers try to capture the profit before someone else can. Even if no one else has noticed the opportunity, the existence of generalised MEV solvers means that as soon as

a profit-capturing transaction is submitted, the MEV solver *will* notice, and the competition will begin all the same. The only known prevention is obfuscation. But, this has not been shown to work in practice, and introduces gas inefficiencies that render arbitrage/liquidation unprofitable. TL;DR, there is always competition when capturing profit.

To win the competition, the keepers must get their transaction to appear before the transactions of their opponents (regardless of whether these opponents are other keepers, or generalised MEV solvers). Generally speaking, higher gas prices result in greater priority for a transaction in the next block. As such, keepers will engage in bidding wars, raising gas prices higher and higher in an attempt to out-bid their opponents. At some point, the next block is mined, or the gas prices are so high that the opportunity is barely profitable, and the bidding ends. The driving of gas prices results in most of the profit being consumed by gas fees. This competitive bidding process is what we call the PGA, because priority in the next block is being auctioned off for gas.

Because gas fees are paid to the miners, they end up winning the majority of available MEV, even though they might not explicitly attempt to extract it.

The Hiding Game

TL;DR: Users route their trades and loans through KeeperDAO, which attempts to extract any arbitrage or liquidation profit available. Those profits are returned back to the user in $ROOK tokens, and profits go into a pool controlled by $ROOK holders.

The *Hiding Game* is the first of three games, and refers to the cooperation between users and keepers to "hide" MEV by wrapping trades/debt in specialised on-chain contracts. These contracts restrict profit extracting opportunities to KeeperDAO itself. This means that keepers in KeeperDAO does *not* have to compete with keepers outside KeeperDAO, and do *not* have to compete with miners.

By giving KeeperDAO priority access to arbitrage and liquidations, the Hiding Game maximises the profits available from these opportunities. This is obviously good for keepers, because it completely removes the gas war. It is also good for Ethereum itself, because it reduces the risk that miners would compromise transaction ordering (or create deep block reorganisations) to extract profit for themselves. But, it is also good for the users.

The profit extracted by KeeperDAO is paid back to the trader: when a trade creates arbitrage, and it is captured by KeeperDAO, the trader is rewarded with freshly minted $ROOK tokens. Similarly, if a loan becomes under-collateralized, and it is liquidated by KeeperDAO, the borrower is also rewarded with freshly minted $ROOK tokens. The $ROOK rewards can be altered by governance so that the value loss to keepers (caused through inflation) is less than what keepers would have spent lost to a gas war. The difference between the value captured, and the value of $ROOK tokens returned to the user is known as the `keeper_fee`.

In this way, KeeperDAO introduces the concept of *hiding*, allowing it to keep trades and lending positions safe from MEV exploitation. These MEV opportunities are under-cut, pre-emotes, and redistributed back to the users of KeeperDAO. For users the incentive is clear: route trades and loans through KeeperDAO, and minimise your losses.

Trading

When a trade is submitted on-chain, there is often profit that can be extracted by front-running and/or tailgating the trade. This is because the trade will cause slippage in its venue, creating arbitrage opportunities against other venues. This is particularly noticeable with AMMs (which are getting more and more popular). Today, keepers are already doing this arbitrage in an uncoordinated way. On one hand, this lack of coordination means that keepers often make close to zero profit, and sometimes even make a loss, thanks to fighting gas wars. On the other hand, the user gets worst-case pricing due to slippage. It is a ***lose-lose*** scenario.

The Hiding Game solves this, by giving priority to KeeperDAO to take trades. Users submit their trades to KeeperDAO (through one of many possible user interfaces), and the keepers analyse the trade to see if profit can be extracted by front-running and/or tailgate the transaction. Keepers within KeeperDAO have priority to take this trade, so if there is profit to be made, the keepers can capitalise on it (in accordance with the rules of the Coordination Game). The profit is returned to the user in the form of freshly minted $ROOK tokens (less the governable `keeper_fee`). Keepers are able to extract more profit than in the uncoordinated scenario, because no gas war needs to be fought. Users get better pricing, because slippage is offset by earning $ROOK. This is now a ***win-win*** scenario.

Today, we already see keepers fighting to front-run/tailgate transactions. Without coordination, keepers are making minimal profit compared to the raw value available, and users are suffering from bad pricing. In this sense, we can see that the Hiding Game is able to offer a strictly better outcome. That is to say, the front-running/tailgating done by KeeperDAO is something users are already exposed to, but without any benefit to themselves. The Hiding Game does not introduce new front-running/tailgating, instead it just returns some of that profit to the user.

Borrowing

Another common on-chain activity is over-collateralised borrowing. Users submit collateral in one token, and receive a smaller loan in another token. Often, users do this to maximise exposure to get leverage, or enter a short position. A common feature in all lending protocols of this nature is liquidation: if the underlying collateral becomes worth too little, then the collateral is auctioned off to the highest bidder, and the user makes a loss.

> *In Compound, most positions require ~150% collateral compared to the loan, and if the collateral drops below ~125%, then the collateral is sold off. The buyer gets 105% of the collateral for 100% of the debt, and the user gets the remainder. The result is a 5% loss for the user, and a small gain for keepers, who find themselves in yet another gas war for the 5% profit (which can sometimes be substantial).*

The Hiding Game allows users to minimise their losses, in exchange for being liquidated slightly earlier. Users can use KeeperDAO to wrap their debt positions, giving KeeperDAO permission to liquidate their collateral slightly earlier than normal (the user can pick how much earlier as a % of their position). The result is similar in nature to the Hiding Game for trades: keepers do not need to fight gas wars for the liquidation, because KeeperDAO has priority, and the profit from the liquidation is returned to the user in the form of $ROOK tokens (less the governable `keeper_fee`). This maximises profit for the keepers, and minimises losses for the user.

The Coordination Game

The Coordination Game is the second of three games introduced by KeeperDAO. It is an economically incentivised game between keepers where coordination is the optimal strategy. The goal is to remove the competitive nature of PGAs. In its place, keepers are encouraged to take turns extracting *all* of the available profit from an MEV opportunity. To incentivise this level of coordination, keepers share their profits with each other. This is possible, because when no PGAs are happening, there is more profit to be shared. All players walk away with more.

To better understand the Coordination Game, it is helpful to look at a concrete example:

> *Alice is an arbitrageur that watches prices on Kyber and Uniswap. One day, she notices that 1 ETH buys 354 USDC on Kyber, and 354 USDC buys 1.01 ETH on Uniswap (after exchange fees). This is clearly an easy way to make 0.01 ETH. To capture this opportunity, Alice submits a transaction with a gas limit of 100,000 and a gas price of 80 GWEI. This results in a transaction cost of 800,000 GWEI. Therefore, if her transaction goes through, Alice will make 0.002 ETH profit after expenses.*
>
> *However, Bob is an arbitrageur that has also seen this opportunity. He has seen Alice's transaction in the mempool and knows that he will need to use a higher gas price if he wants to submit a transaction that will appear before Alice's transaction in the next block. So, Bob submits a transaction with a gas price of 90 GWEI. If his transaction does end up coming first, he will make 0.001 ETH profit after expenses. (For simplicity, we assume that Bob submits a transaction with the same gas limit as Alice.)*
>
> *But, Bob is not the only one watching the mempool. Alice sees Bob's counter-bid and decides to counter his counter-bid by submitting a new transaction with a gas price of 95 GWEI. Her transaction is now likely to come before Bob's, but she is now only standing to make 0.0005 ETH. Still, better than nothing.*
>
> *Again, Bob sees this counter-counter-bid. But, he decides that profitability is now too low (after all, increasing gas prices is only a probabilistic game). Instead, he cancels his transaction. This transaction has a gas price of 91 GWEI (necessary to cancel his previous one), but it only has a gas limit of 21,000.*
>
> *Suddenly, the next block is mined. Alice has won the bid, and she gets her 0.0005 ETH profit. Unfortunately for Bob, his cancellation did not have time to propagate through the mempool and his original counter-bid is still included and he has incurred a loss of 0.009 ETH.*

A lot is missing from this example: arbitrageurs will often use gas tokens to reduce actual gas prices, arbitrageurs cannot always see all transactions in the mempool, and arbitrageurs will play all kinds of games with each other to try and obfuscate what is happening, or to trick each other into making false moves. But the structure of the problem is clear: arbitrageurs are competing and make less profit as gas prices are driven higher. It can even (and in practice, does) result in a loss for the losing bidder. But what happens if we try to optimise by being less competitive?

> *On these kinds of opportunities, Alice sometimes makes 0.0005 ETH, and Bob sometimes makes 0.0005 ETH. It depends on block times, the mempool, and the ever raging arms race of sneaky counter tactics. Alice and Bob are sick and tired of the games and decide to coordinate; Alice and Bob will take turns capturing opportunities and share the profits equally between each other.*
>
> *The same opportunity arises again — 0.01 ETH profit to be made on arbitrage between Kyber and Uniswap — and it is currently Alice's turn to capture any arbitrage that exists. Alice submits the same transaction as previously, but because Bob is not being competitive, she wins with her initial gas price of 80 GWEI. Alice splits the profit with Bob and both arbitrageurs walk away with 0.001 ETH. Alice has made double what she would have made in the competitive scenario, and Bob has avoided incurring any losses (and has even made a profit equal to Alice).*

By avoiding competitiveness, and taking turns, gas prices remain low, and more value can be extracted by the arbitrageurs, even after profit sharing with each other. Although this scenario is very simple, it strikes at the heart of the problem, and is generalizable to more complex situations.

Goals

Before we can describe a general solution, we should develop an understanding of the precise goals of KeeperDAO with respect to the Coordination Game. As KeeperDAO grows and evolves, it should attempt to adhere to these goals to the best that it can. We will call them the *Laws of the Coordination Game*.

1. The profitability of a keeper that is playing the Coordination Game must be higher than when the same keeper is not playing the Coordination Game.
2. The sum profitability of all keepers playing the Coordination Game is higher than sum profitability of the same keepers if they were not playing the Coordination Game.
3. The relative profitability between keepers playing the Coordination Game is the same as the relative profitability between the same keepers if they were not playing the Coordination Game.

Scheduling

Scheduling is the process of deciding which keepers should “go first” when attempting to capture MEV.

The identity of a keeper is created on Ethereum through the KEEPER contract. Anyone can create an identity through this contract by bonding an amount of $ROOK to their identity. Let the amount of $ROOK bonded to keeper `k` in `[0, …, n)` be given as:

```
bond k
```

Every 100 blocks, the schedule is updated. The keeper in the first position is selected randomly, where the relative probability of selection between keepers is equal to the relative weight between keepers, where weight is given as:

```
weight k = sqrt (bond k) / sum (map (sqrt . bond) 0… n)
```

The selected keeper is placed into the first position, removed from future consideration, and the process is repeated for the second position in the schedule. This continues until all keepers with more than some `min_bond_threshold` of $ROOK have been assigned a place in the schedule, where `min_bond_threshold` is a governable parameter.

Keepers can then behave normally: watching for on-chain opportunities, competing with opponents, and so on. The only change required is that keepers should not attempt to compete with other keepers that are higher in the schedule. This encourages keepers to share knowledge about on-chain opportunities (and community practices around doing this will be encouraged, and quickly standardized).

What we have here is an almost perfect analogy to the *iterated prisoner’s dilemma*. It is known that the optimal strategy in such a game is to employ a tit-for-tat strategy: if a keeper competes with you out-of-turn in one iteration, then you should compete with them out-of-turn in the next iteration. Keepers can also vote to remove misbehaving keepers that are continuously competing out-of-turn. Voting power is given by:

```
vote k = weight k
```

Determining whether a keeper has competed out-of-turn can be done using local observations of transactions. If you see a transaction from Bob after you see a transaction from Alice, and the transaction from Bob has a higher gas price than the transaction from Alice, and Alice is higher in the schedule than Bob, you should consider Bob to have competed out-of-turn. If you see a transaction from a keeper that is not playing in the Coordination Game, then you forget all transactions you have seen previously that have a lower gas price.

Note: these game rules allow keepers to try and capture opportunities that another keeper higher in the schedule has missed, or is being too slow to react to. We fully expect keepers to slowly evolve to act with the appropriate delay when they are not first in the schedule (such a delay will very likely converge to a value that is negligible compared to block times).

Profit Sharing

The performance of a keeper is measured by the amount of profit that it returns to the shared liquidity pool. At the end of every day, $ROOK is minted and distributed to the keepers in equal value to the amount profit returned (less a governable `liquidity_fee`). The $ROOK is distributed proportionally to the amount of $ROOK bonded.

Performance of keeper `k` is given by:

```
perf k = profit k + sqrt (gas_spent k)
```

Including the `gas_spent` when computing the performance of a keeper ensures that keepers are incentivized to battle with opponents that are not currently part of the Coordination Game. By taking the `sqrt` of the `gas_spent`, keepers are still dominantly incentivized to return concrete profits wherever possible (instead of always grim-triggering, even when it may not be necessary).

At the end of every day, the performance of a keeper is compared against its bond. If the relative performance of a keeper is lower than its relative bond, the keeper is said to have under-performed and it loses some of its bond. If the relative performance of a keeper is higher than its relative bond, the keeper is said to have over-performed and it is rewarded with the bond lost by others.

```
relative_perf k = (perf k / sum (map perf 0… n))
relative_bond k = (bond k / sum (map bond 0… n))
reward k = log (bond k) * (relative_perf k - relative_bond k)
```

This burns and mints $ROOK for keepers that are under and over performing respectively. The `reward k` is automatically added/deducted to/from the bond of the keeper. This ensures that, over time, the distribution of $ROOK to the keepers converges to the distribution of their performance. In turn, this results in the scheduling of keepers to favour higher performing keepers, and keepers that are not active/participating will eventually find themselves dropped from the schedule.

The Incentive Game

The third and final game is the *Incentive Game*. This game exists to encourage specific user behaviours, and to encourage active governance.

$ROOK

The $ROOK token is at the heart of the Incentive Game. In the Hiding Game and the Coordination Game (see the previous two sections) we have already discussed two circumstances where $ROOK is minted and distributed.

1. When users in the Hiding Game create profit that is captured by KeeperDAO, some of that profit is paid back to the user in freshly minted $ROOK.
2. When keepers return profit during the Coordination Game, some of that profit is paid back to the keepers in freshly minted $ROOK.

But, there are other scenarios where it is useful to mint $ROOK. In particular:

1. To encourage users to provide liquidity for KeeperDAO to use to underwrite the Hiding Game, and to leverage in the Coordination Game.
2. To encourage users to adopt the Hiding Game by routing trades and debt positions through it.
3. To encourage keepers to join the Coordination Game and work with other keepers in KeeperDAO, instead of competing.
4. To encourage active participation in governance.

Throughout the lifetime of KeeperDAO, $ROOK will be minted inline with a preset emission schedule. A portion of the minted $ROOK will be paid out to each of the four activities mentioned above. See Governance below for more information about how $ROOK is controlled.

Governance

Governance is a complex and tricky topic for every decentralized protocol. In KeeperDAO, $ROOK is used for governance. The exact mechanism of governance is still to be decided, but will be a completely on-chain process where code is deployed/executed/upgraded as an automatic and direct consequence of voting.

Managing Profit

Whether it is through the Hiding Game, or the Coordination Game, KeeperDAO collects profit. All of this profit is stored in a FeePool that will be governed on-chain by $ROOK holders, who get to vote on how those funds are used. This could be to fund open-source research and development in relevant fields, incentivize participation and integration, or even distribute the fees back to $ROOK holders.

Managing Parameters

KeeperDAO has many parameters. The shape of the $ROOK emission curve, the distribution of emissions to different incentive programs, the `keeper_fee`, the `liquidity_fee`, etc. All of these parameters will be governed on-chain by the $ROOK token holders.

Managing Upgrades

As the landscape changes, so too must KeeperDAO change. Unsurprisingly, it is also the responsibility of $ROOK holders to propose and instantiate such changes using on-chain governance.

Future

KeeperDAO will be ever evolving. Some of the features that are currently under development, and will be added in future iterations:

- The ability for $ROOK holders to delegate their $ROOK to keepers to encourage/discourage behaviours that cannot be automatically captured by on-chain smart contracts (for example: ensuring that the keepers are grim-triggering external opponents, and ensuring keepers do not “act out of turn”).
- The ability for KeeperDAO to front-run attacks against DeFi protocols and return the funds safely back to the appropriate parties.
- Continued development into new adapters for the Hiding Game to include support for as many DeFi protocols as possible.
- Continued research into general MEV solvers for themselves, to capture as much opportunity as possible.

There is much thinking, much work, and much more discussion and collaboration. Get involved, and **help evolve the protocol**.