The 20 DeFi Whitepapers

A curated set of the most relevant whitepapers

Curated & Compiled By
Jamiel Sheikh
Raymond Chen

chainhaus

Protocols Included

Maker
Compound
AAVE
Curve
Uniswap
Instadapp
Synthetix
Balancer
Bancor
Vesper
RenVM
Flexa
Harvest Finance
Tornado Cash
Nexus Mutual
CREAM Finance
dYdX
KEEP Network
1INCH

The Maker Protocol: MakerDAO's Multi-Collateral Dai (MCD) System

Abstract

The Maker Protocol, also known as the Multi-Collateral Dai (MCD) system, allows users to generate Dai by leveraging collateral assets approved by "Maker Governance." Maker Governance is the community organized and operated process of managing the various aspects of the Maker Protocol. Dai is a decentralized, unbiased, collateral-backed cryptocurrency soft-pegged to the US Dollar. Resistant to hyperinflation due to its low volatility, Dai offers economic freedom and opportunity to anyone, anywhere.

This white paper is a reader-friendly description of the Protocol, which is built on the Ethereum blockchain. Technically savvy users might want to head directly to Introduction to the Maker Protocol in the Maker Documentation Portal for an in-depth explanation of the entire system.

About MakerDAO

MakerDAO is an open-source project on the Ethereum blockchain and a Decentralized Autonomous Organization[1] created in 2014. The project is managed by people around the world who hold its governance token, MKR.

[1] Note that Decentralized Autonomous Organizations, or DAOs, are understood in the Ethereum community as largely social and technical communities centered around a particular mission or project, and does not necessarily imply the existence of traditional corporate forms.

Through a system of scientific governance involving Executive Voting and Governance Polling, MKR holders manage the Maker Protocol and the financial risks of Dai to ensure its stability, transparency, and efficiency. MKR voting weight is proportional to the amount of MKR a voter stakes in the voting contract, DSChief. In other words, the more MKR tokens locked in the contract, the greater the voter's decision-making power.

About the Maker Protocol

The Maker Protocol, built on the Ethereum blockchain,[2] enables users to create currency. Current elements of the Maker Protocol are the Dai stablecoin, Maker Collateral Vaults, Oracles, and Voting. MakerDAO governs the Maker Protocol by deciding on key parameters (e.g., stability fees, collateral types/rates, etc.) through the voting power of MKR holders.

The Maker Protocol, one of the largest decentralized applications (dapps) on the Ethereum blockchain, was the first decentralized finance (DeFi) application to earn significant adoption.

About the Maker Foundation

The Maker Foundation, which is part of the global Maker community, built and launched the Maker Protocol in conjunction with a number of outside partners. It is currently working with the MakerDAO community to bootstrap decentralized governance of the project and drive it toward complete decentralization.

About the Dai Foundation

The Dai Foundation, based in Denmark, is self-governing and independent of the Maker Foundation. It was formed to house the Maker community's key intangible assets, such as trademarks and code copyrights, and it operates solely on the basis of objective and rigid statutes that define its mandate. Its purpose, as noted in the Dai Foundation Trust Deed, is to safeguard what cannot be technologically decentralized in the Maker Protocol.

[2] https://ethereum.org/

Introduction

Beginning in 2015, the MakerDAO project operated with developers around the globe working together on the first iterations of code, architecture, and documentation. In December 2017, the first MakerDAO formal white paper was published, introducing the original Dai (now Sai) Stablecoin System.

The white paper described how anyone could generate Dai using that system by leveraging Ethereum (ETH) as collateral through unique smart contracts known as Collateralized Debt Positions (CDPs). Given that ETH was the only collateral asset accepted by the system, the Dai generated was called Single-Collateral Dai (SCD), or Sai. That white paper also included a plan to upgrade the system to support multiple collateral asset types in addition to ETH. What was then an intention, became a reality in November 2019.

The Dai Stablecoin System, today called the Maker Protocol, now accepts as collateral any Ethereum-based asset that has been approved by MKR holders, who also vote on corresponding Risk Parameters for each collateral asset. Voting is a critical component of the Maker decentralized governance process.

Welcome to **Multi-Collateral Dai (MCD)**.

In MCD We Trust

Blockchain technology provides an unprecedented opportunity to ease the public’s growing frustration with—and distrust of—dysfunctional centralized financial systems. By distributing data across a network of computers, the technology allows any group of individuals to embrace transparency rather than central-entity control. The result is an unbiased, transparent, and highly efficient permissionless system—one that can improve current global financial and monetary structures and better serve the public good.

Bitcoin was created with this goal in mind. But, while Bitcoin succeeds as a cryptocurrency on a number of levels, it is not ideal as a medium of exchange because its fixed supply and speculative nature results in volatility, which prevents it from proliferating as mainstream money.

The Dai stablecoin, on the other hand, succeeds where Bitcoin fails precisely because Dai is designed to *minimize price volatility.* A decentralized, unbiased, collateral-backed cryptocurrency that is soft-pegged to the US Dollar, Dai's value is in its stability.

Since the release of Single-Collateral Dai in 2017, user adoption of the stablecoin has risen dramatically, and it has become a building block for decentralized applications that help expand the DeFi (decentralized finance) movement. Dai's success is part of a wider industry movement for stablecoins, which are cryptocurrencies designed to maintain price value and function like money.

For example, in February 2019, JPMorgan became the first bank in the United States to create and test a digital coin that represents 1 USD.[3] As the cryptocurrency industry grows, other banks, financial services companies, and even governments will create stable digital currencies (e.g., Central Bank Digital Currencies), as will large organizations outside of the finance sector. Facebook, for example, announced its plans for Libra, "a stable digital cryptocurrency that will be fully backed by a reserve of real assets,"[4] in June 2019. However, such proposals forfeit the core value proposition of blockchain technology: global adoption of a common infrastructure without a central authority or administrator that may abuse its influence.

An Overview of the Maker Protocol and Its Features

The Maker Protocol

The Maker Protocol is one of the largest dapps on the Ethereum blockchain. Designed by a disparate group of contributors, including developers within

[3] https://www.jpmorgan.com/global/news/digital-coin-payments
[4] https://libra.org/en-US/wp-content/uploads/sites/23/2019/06/LibraWhitePaper_en_US.pdf

the Maker Foundation, its outside partners, and other persons and entities, it is the first decentralized finance (DeFi) application to see significant adoption.

The Maker Protocol is managed by people around the world who hold its governance token, MKR. Through a system of scientific governance involvingExecutive Voting and Governance Polling, MKR holders govern the Protocol and the financial risks of Dai to ensure its stability, transparency, and efficiency. One MKR token locked in a voting contract equals one vote.

The Dai Stablecoin

The Dai stablecoin is a decentralized, unbiased, collateral-backed cryptocurrency soft-pegged to the US Dollar. Dai is held in cryptocurrency wallets or within platforms, and is supported on Ethereum and other popular blockchains.

Dai is easy to generate, access, and use. Users generate Dai by depositing collateral assets into Maker Vaults within the Maker Protocol. This is how Dai is entered into circulation and how users gain access to liquidity. Others obtain Dai by buying it from brokers or exchanges, or simply by receiving it as a means of payment.

Once generated, bought, or received, Dai can be used in the same manner as any other cryptocurrency: it can be sent to others, used as payments for goods and services, and even held as savings through a feature of the Maker Protocol called the Dai Savings Rate (DSR).

Every Dai in circulation is directly backed by excess collateral, meaning that the value of the collateral is higher than the value of the Dai debt, and all Dai transactions are publicly viewable on the Ethereum blockchain.

What Properties of Dai Function Similarly to Money?

Generally, money has four functions:

1. A store of value
2. A medium of exchange

3. A unit of account
4. A standard of deferred payment

Dai has properties and use cases designed to serve these functions.

Dai as a Store of Value

A store of value is an asset that keeps its value without significant depreciation over time. Because Dai is a stablecoin, it is designed to function as a store of value even in a volatile market.

Dai as a Medium of Exchange

A medium of exchange is anything that represents a standard of value and is used to facilitate the sale, purchase, or exchange (trade) of goods or services. The Dai stablecoin is used around the world for all types of transactional purposes.

Dai as a Unit of Account

A unit of account is a standardized measurement of value used to price goods and services (e.g., USD, EUR, YEN). Currently, Dai has a target price of 1USD (1 Dai = 1 USD). While Dai is not used as a standard measurement of value in the off-chain world, it functions as a unit of account within the Maker Protocol and some blockchain dapps, whereby Maker Protocol accounting or pricing of dapp services is in Dai rather than a fiat currency like USD.

Dai as a Standard of Deferred Payment

Dai is used to settle debts within the Maker Protocol (e.g., users use Dai to pay the stability fee and close their Vaults). This benefit separates Dai from other stablecoins.

Collateral Assets

Dai is generated, backed, and kept stable through collateral assets that are deposited into Maker Vaults on the Maker Protocol. A collateral asset is a digital asset that MKR holders have voted to accept into the Protocol.

To generate Dai, the Maker Protocol accepts as collateral any Ethereum-based asset that has been approved by MKR holders. MKR holders also must also approve specific, corresponding Risk Parameters for each accepted collateral (e.g., more stable assets might get more lenient Risk Parameters, while more risky assets could get stricter Risk Parameters). Detailed information on Risk Parameters is below. These and other decisions of MKR holders are made through the Maker decentralized governance process.

Maker Vaults

All accepted collateral assets can be leveraged to generate Dai in the Maker Protocol through smart contracts called Maker Vaults. Users can access the Maker Protocol and create Vaults through a number of different user interfaces (i.e., network access portals), including Oasis Borrow and various interfaces built by the community. Creating a Vault is not complicated, but generating Dai does create an obligation to repay the Dai, along with a Stability Fee, in order to withdraw the collateral leveraged and locked inside a Vault.

Vaults are inherently non-custodial: Users interact with Vaults and the Maker Protocol directly, and each user has complete and independent control over their deposited collateral as long the value of that collateral doesn't fall below the required minimum level (the Liquidation Ratio, discussed in detail below).

Interacting with a Maker Vault

- **Step 1: Create and Collateralize a Vault**
  A user creates a Vault via the Oasis Borrow portal or a community-created interface, such as Instadapp, Zerion, or MyEtherWallet, by funding it with a specific type and amount of collateral that will be used to generate Dai. Once funded, a Vault is considered collateralized.
- **Step 2: Generate Dai from the Collateralized Vault**
  The Vault owner initiates a transaction, and then confirms it in her unhosted cryptocurrency wallet in order to generate a specific amount of Dai in exchange for keeping her collateral locked in the Vault.
- **Step 3: Pay Down the Debt and the Stability Fee**
  To retrieve a portion or all of the collateral, a Vault owner must pay down or completely pay back the Dai she generated, plus the Stability

Fee that continuously accrues on the Dai outstanding. The Stability Fee can only be paid in Dai.

- **Step 4: Withdraw Collateral**
  With the Dai returned and the Stability Fee paid, the Vault owner can withdraw all or some of her collateral back to her wallet. Once all Dai is completely returned and all collateral is retrieved, the Vault remains empty until the owner chooses to make another deposit.

Importantly, each collateral asset deposited requires its own Vault. So, some users will own multiple Vaults with different types of collateral and levels of collateralization.

Liquidation of Risky Maker Vaults

To ensure there is always enough collateral in the Maker Protocol to cover the value of all outstanding debt (the amount of Dai outstanding valued at the Target Price), any Maker Vault deemed too risky (according to parameters established by Maker Governance) is liquidated through automated Maker Protocol auctions. The Protocol makes the determination after comparing the Liquidation Ratio to the current collateral-to-debt ratio of a Vault. Each Vault type has its own Liquidation Ratio, and each ratio is determined by MKR voters based on the risk profile of the particular collateral asset type.

Maker Protocol Auctions

The auction mechanisms of the Maker Protocol enable the system to liquidate Vaults even when price information for the collateral is unavailable. At the point of liquidation, the Maker Protocol takes the liquidated Vault collateral and subsequently sells it using an internal market-based auction mechanism. This is a **Collateral Auction**.

The Dai received from the Collateral Auction is used to cover the Vault's outstanding obligations, including payment of the Liquidation Penalty fee set by MKR voters for that specific Vault collateral type.

If enough Dai is bid in the Collateral Auction to fully cover the Vault obligations plus the Liquidation Penalty, that auction converts to a **Reverse Collateral Auction** in an attempt to sell as little collateral as possible. Any leftover collateral is returned to the original Vault owner.

If the Collateral Auction does not raise enough Dai to cover the Vault's outstanding obligation, the deficit is converted into Protocol debt. Protocol debt is covered by the Dai in the Maker Buffer. If there is not enough Dai in the Buffer, the Protocol triggers a **Debt Auction**. During a Debt Auction, MKR is minted by the system (increasing the amount of MKR in circulation), and then sold to bidders for Dai.

Dai proceeds from the Collateral Auction go into the Maker Buffer, which serves as a buffer against an increase of MKR overall supply that could result from future uncovered Collateral Auctions and the accrual of the Dai Savings Rate (discussed in detail below).

If Dai proceeds from auctions and Stability Fee payments exceed the Maker Buffer limit (a number set by Maker Governance), they are sold through a **Surplus Auction**. During a Surplus Auction, bidders compete by bidding decreasing amounts of MKR to receive a fixed amount of Dai. Once the Surplus Auction has ended, the Maker Protocol autonomously destroys the MKR collected, thereby reducing the total MKR supply.

Example (Collateral Auction Process):

A large Vault becomes undercollateralized due to market conditions. An Auction Keeper then detects the undercollateralized Vault opportunity and initiates liquidation of the Vault, which kicks off a Collateral Auction for, say, 50 ETH.

Each Auction Keeper has a **bidding model** to assist in winning auctions. A bidding model includes a price at which to bid for the collateral (ETH, in this example). The Auction Keeper uses the token price from its bidding model as the basis for its bids in the first phase of a Collateral Auction, where increasing Dai bids are placed for the set amount of collateral. This amount represents the price of the total Dai wanted from the collateral auction.

Now, let's say the Auction Keeper bids 5,000 Dai for the 50 ETH to meet this amount. The Dai bid is transferred from the Vault Engine to the Collateral Auction contract. With enough Dai in the Collateral Auction contract to cover the system's debt plus the Liquidation Penalty, the first phase of the Collateral Auction is over.

In order to reach the price defined in its bidding model, the Auction Keeper submits a bid in the second phase of the Collateral Auction. In this phase, the objective is to return as much of the collateral to the Vault owner as the market will allow. The bids that the Auction Keepers place are for fixed Dai amounts and decreasing amounts of ETH. For instance, the bidding model of the Keeper in this example seeks a bid price of 125 Dai per ETH, so it offers 5000 Dai for 40 ETH. Additional Dai for this bid is transferred from the Vault Engine to the Collateral Auction contract. After the bid duration limit is reached and the bid expires, the Auction Keeper claims the winning bid and settles the completed Collateral Auction by collecting the won collateral.

Key External Actors

In addition to its smart contract infrastructure, the Maker Protocol involves groups of external actors to maintain operations: Keepers, Oracles, and Global Settlers (Emergency Oracles), and Maker community members. Keepers take advantage of the economic incentives presented by the Protocol; Oracles and Global Settlers are external actors with special permissions in the system assigned to them by MKR voters; and Maker community members are individuals and organizations that provide services.

Keepers

A Keeper is an independent (usually automated) actor that is incentivized by arbitrage opportunities to provide liquidity in various aspects of a decentralized system. In the Maker Protocol, Keepers are market participants that help Dai maintain its Target Price ($1): they sell Dai when the market price is above the Target Price, and buy Dai when the market price is below the Target Price. Keepers participate in Surplus Auctions, Debt Auctions, and Collateral Auctions when Maker Vaults are liquidated.

Price Oracles

The Maker Protocol requires real-time information about the market price of the collateral assets in Maker Vaults in order to know when to trigger Liquidations.

The Protocol derives its internal collateral prices from a decentralized Oracle infrastructure that consists of a broad set of individual nodes called Oracle Feeds. MKR voters choose a set of trusted Feeds to deliver price information to the system through Ethereum transactions. They also control how many Feeds are in the set.

To protect the system from an attacker attempting to gain control of a majority of the Oracles, the Maker Protocol receives price inputs through the Oracle Security Module (OSM), not from the Oracles directly. The OSM, which is a layer of defense between the Oracles and the Protocol, delays a price for one hour, allowing Emergency Oracles or a Maker Governance vote to freeze an Oracle if it is compromised. Decisions regarding Emergency Oracles and the price delay duration are made by MKR holders.

Emergency Oracles

Emergency Oracles are selected by MKR voters and act as a last line of defense against an attack on the governance process or on other Oracles. Emergency Oracles are able to freeze individual Oracles (e.g., ETH and BAT Oracles) to mitigate the risk of a large number of customers trying to withdraw their assets from the Maker Protocol in a short period of time, as they have the authority to unilaterally trigger an Emergency Shutdown.

DAO Teams

DAO teams consist of individuals and service providers, who may be contracted through Maker Governance to provide specific services to MakerDAO. Members of DAO teams are independent market actors and are not employed by the Maker Foundation.

The flexibility of Maker Governance allows the Maker community to adapt the DAO team framework to suit the services needed by the ecosystem based on real-world performance and emerging challenges.

Examples of DAO team member roles are the Governance Facilitator, who supports the communication infrastructure and processes of governance, and Risk Team members, who support Maker Governance with financial risk research and draft proposals for onboarding new collateral and regulating existing collateral.

While the Maker Foundation has bootstrapped Maker Governance to date, it is anticipated that the DAO will take full control, conduct MKR votes, and fill these varied DAO team roles in the near future.

The Dai Savings Rate

The Dai Savings Rate (DSR) allows any Dai holder to earn savings automatically and natively by locking their Dai into the DSR contract in the Maker Protocol. It can be accessed via the Oasis Save portal or through various gateways into the Maker Protocol. Users aren't required to deposit a minimum amount to earn the DSR, and they can withdraw any or all of their Dai from the DSR contract at any time.

The DSR is a global system parameter that determines the amount Dai holders earn on their savings over time. When the market price of Dai deviates from the Target Price due to changing market dynamics, MKR holders can mitigate the price instability by voting to modify the DSR accordingly:

- If the market price of Dai is above 1 USD, MKR holders can choose to gradually decrease the DSR, which will reduce demand and should reduce the market price of Dai toward the 1 USD Target Price.
- If the market price of Dai is below 1 USD, MKR holders can choose to gradually increase the DSR, which will stimulate demand and should increase the market price of Dai toward the 1 USD Target Price.

Initially, adjustment of the DSR will depend on a weekly process, whereby MKR holders first evaluate and discuss public market data and proprietary data provided by market participants, and then vote on whether an adjustment is necessary or not. The long-term plan includes implementation of the DSR Adjustment Module, an Instant Access Module that directly controls both the DSR and the Base Rate. This module allows for easy adjustment of the DSR (within strict size and frequency boundaries set by MKR holders) by an MKR holder on behalf of the larger group of MKR holders. The motivation behind this plan is to enable nimble responses to rapidly changing market conditions, and to avoid overuse of the standard governance process ofExecutive Voting and Governance Polling.

Governance of the Maker Protocol

Use of the MKR Token in Maker Governance

The MKR token—the governance token of the Maker Protocol—allows those who hold it to *vote* on changes to the Maker Protocol. Note that anyone, not only MKR holders, can *submit* proposals for an MKR vote.

Any voter-approved modifications to the governance variables of the Protocol will likely not take effect immediately in the future; rather, they could be delayed by as much as 24 hours if voters choose to activate the Governance Security Module (GSM). The delay would give MKR holders the opportunity to protect the system, if necessary, against a malicious governance proposal (e.g., a proposal that alters collateral parameters contrary to established monetary policies or that allows for security mechanisms to be disabled) by triggering a Shutdown.

Polling and Executive Voting

In practice, the Maker Governance process includes proposal polling and Executive Voting. Proposal polling is conducted to establish a rough consensus of community sentiment before any Executive Votes are cast. This helps to ensure that governance decisions are considered throughtfully and reached by consensus prior to the voting process itself. Executive Voting is held to approve (or not) changes to the state of the system. An example of an Executive Vote could be a vote to ratify Risk Parameters for a newly accepted collateral type.

At a technical level, smart contracts manage each type of vote. A Proposal Contract is a smart contract with one or more valid governance actions programmed into it. It can only be executed once. When executed, it immediately applies its changes to the internal governance variables of the Maker Protocol. After execution, the Proposal Contract cannot be reused.

Any Ethereum Address can deploy valid Proposal Contracts. MKR token holders can then cast approval votes for the proposal that they want to elect as the Active Proposal. The Ethereum address that has the highest number of approval votes is elected as the Active Proposal. The Active Proposal is empowered to gain administrative access to the internal governance variables of the Maker Protocol, and then modify them.

The MKR Token's Role in Recapitalization
In addition to its role in Maker Governance, the MKR token has a complementary role as the recapitalization resource of the Maker Protocol. If the system debt exceeds the surplus, the MKR token supply may increase through a Debt Auction (see above) to recapitalize the system. This risk inclines MKR holders to align and responsibly govern the Maker ecosystem to avoid excessive risk-taking.

MKR Holder Responsibilities
MKR holders can vote to do the following:

- Add a new collateral asset type with a unique set of Risk Parameters.
- Change the Risk Parameters of one or more existing collateral asset types, or add new Risk Parameters to one or more existing collateral asset types.
- Modify the Dai Savings Rate.
- Choose the set of Oracle Feeds.
- Choose the set of Emergency Oracles.
- Trigger Emergency Shutdown.
- Upgrade the system.

MKR holders can also allocate funds from the Maker Buffer to pay for various infrastructure needs and services, including Oracle infrastructure and collateral risk management research. The funds in the Maker Buffer are revenues from Stability Fees, Liquidation Fees, and other income streams.

The governance mechanism of the Maker Protocol is designed to be as flexible as possible, and upgradeable. Should the system mature under the guidance of the community, more advanced forms of Proposal Contracts could, in theory, be used, including Proposal Contracts that are bundled. For example, one proposal contract may contain both an adjustment of a Stability Fee and an adjustment of the DSR. Nonetheless, those revisions will remain for MKR holders to decide.

Risk Parameters Controlled by Maker Governance

Each Maker Vault type (e.g., ETH Vault and BAT Vault) has its own unique set of Risk Parameters that enforce usage. The parameters are determined based on the risk profile of the collateral, and are directly controlled by MKR holders through voting.

The Key Risk Parameters for Maker Vaults are:

- **Debt Ceiling**: A Debt Ceiling is the maximum amount of debt that can be created by a single collateral type. Maker Governance assigns every collateral type a Debt Ceiling, which is used to ensure sufficient diversification of the Maker Protocol collateral portfolio. Once a collateral type has reached its Debt Ceiling, it becomes impossible to create more debt unless some existing users pay back all or a portion of their Vault debt.
- **Stability Fee**: The Stability Fee is an annual percentage yield calculated on top of how much Dai has been generated against a Vault's collateral. The fee is paid in Dai only, and then sent into the Maker Buffer.
- **Liquidation Ratio**: A low Liquidation Ratio means Maker Governance expects low price volatility of the collateral; a high Liquidation Ratio means high volatility is expected.
- **Liquidation Penalty**: The Liquidation Penalty is a fee added to a Vault's total outstanding generated Dai when a Liquidation occurs. The Liquidation Penalty is used to encourage Vault owners to keep appropriate collateral levels.
- **Collateral Auction Duration**: The maximum duration of Collateral auctions is specific to Maker Vaults. Debt and Surplus auction durations are global system parameters.
- **Auction Bid Duration:** Amount of time before an individual bid expires and closes the auction.
- **Auction Step Size**: This Risk Parameter exists to incentivize early bidders in auctions, and prevent abuse by bidding a tiny amount above an existing bid.

Risk and Mitigation Responsibilities of Governance

The successful operation of the Maker Protocol depends on Maker Governance taking necessary steps to mitigate risks. Some of those risks are identified below, each followed by a mitigation plan.

A malicious attack on the smart contract infrastructure by a bad actor.
One of the greatest risks to the Maker Protocol is a malicious actor—a programmer, for example, who discovers a vulnerability in the deployed smart contracts, and then uses it to break the Protocol or steal from it.

In the worst-case scenario, all decentralized digital assets held as collateral in the Protocol are stolen, and recovery is impossible.

Mitigation: The Maker Foundation's highest priority is the security of the Maker Protocol, and the strongest defense of the Protocol is Formal Verification. The Dai codebase was the first codebase of a decentralized application to be formally verified.

In addition to formal system verification, contracted security audits by the best security organizations in the blockchain industry, third-party (independent) audits, and bug bounties are part of the Foundation’s security roadmap. To review the formal verification report and various Maker Protocol audits, visit Maker’s Multi-Collateral Dai Security Github repository.

These security measures provide a strong defense system; however, they are not infallible. Even with formal verification, the mathematical modeling of intended behaviors may be incorrect, or the assumptions behind the intended behavior itself may be incorrect.

A black swan event
A black swan event is a rare and critical surprise attack on a system. For the Maker Protocol, examples of a black swan event include:

- An attack on the collateral types that back Dai.
- A large, unexpected price decrease of one or more collateral types.
- A highly coordinated Oracle attack.

- A malicious Maker Governance proposal.

Please note that this list of potential "black swans" is not exhaustive and not intended to capture the extent of such possibilities.

Mitigation: While no one solution is failsafe, the careful design of the Maker Protocol (the Liquidation Ratio, Debt Ceilings, the Governance Security Module, the Oracle Security Module, Emergency Shutdown, etc.) in conjunction with good governance (e.g., swift reaction in a crisis, thoughtful risk parameters, etc.) help to prevent or mitigate potentially severe consequences of an attack.

Unforeseen pricing errors and market irrationality
Oracle price feed problems or irrational market dynamics that cause variations in the price of Dai for an extended period of time can occur. If confidence in the system is lost, rate adjustments or even MKR dilution could reach extreme levels and still not bring enough liquidity and stability to the market.

Mitigation: Maker Governance incentivizes a sufficiently large capital pool to act as Keepers of the market in order to maximize rationality and market efficiency, and allow the Dai supply to grow at a steady pace without major market shocks. As a last resort, Emergency Shutdown can be triggered to release collateral to Dai holders, with their Dai claims valued at the Target Price.

User Abandonment for Less Complicated Solutions
The Maker Protocol is a complex decentralized system. As a result of its complexity, there is a risk that inexperienced cryptocurrency users will abandon the Protocol in favor of systems that may be easier to use and understand.

Mitigation: While Dai is easy to generate and use for most crypto enthusiasts and the Keepers that use it for margin trading, newcomers might find the Protocol difficult to understand and navigate. Although Dai is designed in such a way that users need not comprehend the underlying mechanics of the Maker Protocol in order to benefit from it, the documentation and numerous resources consistently provided by the Maker community and the Maker Foundation help to ensure onboarding is as uncomplicated as possible.

Dissolution of The Maker Foundation
The Maker Foundation currently plays a role, along with independent actors, in maintaining the Maker Protocol and expanding its usage worldwide, while facilitating Governance. However, the Maker Foundation plans to dissolve once MakerDAO can manage Governance completely on its own. Should MakerDAO fail to sufficiently take the reins upon the Maker Foundation's dissolution, the future health of the Maker Protocol could be at risk.

Mitigation: MKR holders are incentivized to prepare for the Foundation's dissolution after it completes "gradual decentralization" of the project. Moreover, successful management of the system should result in sufficient funds for governance to allocate to the continued maintenance and improvement of the Maker Protocol.

General Issues with Experimental Technology

Users of the Maker Protocol (including but not limited to Dai and MKR holders) understand and accept that the software, technology, and technical concepts and theories applicable to the Maker Protocol are still unproven and there is no warranty that the technology will be uninterrupted or error-free. There is an inherent risk that the technology could contain weaknesses, vulnerabilities, or bugs causing, among other things, the complete failure of the Maker Protocol and/or its component parts.

Mitigation: See “A malicious attack on the smart contract infrastructure by a bad actor” above. The Mitigation section there explains the technical auditing in place to ensure the Maker Protocol functions as intended.

Price Stability Mechanisms

The Dai Target Price

The Dai Target Price is used to determine the value of collateral assets Dai holders receive in the case of an Emergency Shutdown. The Target Price for Dai is 1 USD, translating to a 1:1 USD soft peg.

Emergency Shutdown

Emergency Shutdown (or, simply, Shutdown) serves two main purposes. First, it is used during emergencies as a last-resort mechanism to protect the Maker Protocol against attacks on its infrastructure and directly enforce the Dai Target Price. Emergencies could include malicious governance actions, hacking, security breaches, and long-term market irrationality. Second, Shutdown is used to facilitate a Maker Protocol system upgrade. The Shutdown process can only be controlled by Maker Governance.

MKR voters are also able to instantly trigger an Emergency Shutdown by depositing MKR into the Emergency Shutdown Module (ESM), if enough MKR voters believe it is necessary. This prevents the Governance Security Module (if active) from delaying Shutdown proposals before they are executed. With Emergency Shutdown, the moment a quorum is reached, the Shutdown takes effect with no delay.

There are three phases of Emergency Shutdown:

1. **The Maker Protocol shuts down; Vault owners withdraw assets.** When initiated, Shutdown prevents further Vault creation and manipulation of existing Vaults, and freezes the Price Feeds. The frozen feeds ensure that all users are able to withdraw the net value of assets to which they are entitled. Effectively, it allows Maker Vault owners to immediately withdraw the collateral in their Vault that is not actively backing debt.
2. **Post-Emergency Shutdown auction processing**
   After Shutdown is triggered, Collateral Auctions begin and must be completed within a specific amount of time. That time period is determined by Maker Governance to be slightly longer than the duration of the longest Collateral Auction. This guarantees that no auctions are outstanding at the end of the auction processing period.
3. **Dai holders claim their remaining collateral**
   At the end of the auction processing period, Dai holders use their Dai to claim collateral directly at a fixed rate that corresponds to the calculated value of their assets based on the Dai Target Price. For example, if the ETH/USD Price Ratio is 200, and a user holds 1000 Dai at the Target Price of 1 USD when Emergency Shutdown is activated,

The user will be able to claim exactly 5 ETH from the Maker Protocol after the auction processing period. There is no time limit for when a final claim can be made. Dai holders will get a proportional claim to each collateral type that exists in the collateral portfolio. Note that Dai holders could be at risk of a haircut, whereby they do not receive the full value of their Dai holdings at the Target Price of 1 USD per Dai. This is due to risks related to declines in collateral value and to Vault owners having the right to retrieve their excess collateral before Dai holders may claim the remaining collateral. For more detailed information on Emergency Shutdown, including the claim priorities that would occur as a result, see the published community documentation.

The Future of the Maker Protocol: Increased Adoption and Full Decentralization

Addressable Market

A cryptocurrency with price stability serves as an important medium of exchange for many decentralized applications. As such, the potential market for Dai is at least as large as the entire decentralized blockchain industry. But the promise of Dai extends well beyond that into other industries.

The following is a non-exhaustive list of current and immediate markets for the Dai stablecoin:

- **Working capital, hedging, and collateralized leverage**. Maker Vaults allow for permissionless trading by users, who can use the Dai generated against Vault collateral for working capital. To date, there have been numerous instances where Vault owners use their Dai to buy additional ETH (same asset as their collateral), thereby creating a leveraged but fully collateralized position.
- **Merchant receipts, cross-border transactions, and remittances**. Foreign exchange volatility mitigation and a lack of intermediaries mean the transaction costs of international trade are significantly reduced when using Dai.
- **Charities and NGOs** when using transparent distributed ledger technology.

- **Gaming.** For blockchain game developers, Dai is the currency of choice. With Dai, game developers integrate not only a currency, but also an entire economy. The composability of Dai allows games to create new player behavior schemes based around decentralized finance.
- **Prediction markets**. Using a volatile cryptocurrency when making an unrelated prediction only increases one's risk when placing the bet. Long-term bets become especially infeasible if the bettor must also gamble on the future price of the volatile asset used to place the bet. That said, the Dai stablecoin would be a natural choice for use in prediction markets.

Asset Expansion

Should MKR holders approve new assets as collateral, those assets will be subject to the same risk requirements, parameters, and safety measures as Dai (e.g., Liquidation Ratios, Stability Fees, Savings Rates, Debt Ceilings, etc.).

Evolving Oracles

MakerDAO was the first project to run reliable Oracles on the Ethereum blockchain. As a result, many decentralized applications use MakerDAO Oracles to ensure the security of their systems and to provide up-to-date price data in a robust manner. This confidence in MakerDAO and the Maker Protocol means that Maker Governance can expand the core Oracle infrastructure service to better suit the needs of decentralized applications.

Conclusion

The Maker Protocol allows users to generate Dai, a stable store of value that lives entirely on the blockchain. Dai is a decentralized stablecoin that is not issued or administered by any centralized actor or trusted intermediary or counterparty. It is unbiased and borderless —available to anyone, anywhere.

All Dai is backed by a surplus of collateral that has been individually escrowed into audited and publicly viewable Ethereum smart contracts.

Anyone with an internet connection can monitor the health of the system anytime at daistats.com.

With hundreds of partnerships and one of the strongest developer communities in the cryptocurrency space, MakerDAO has become the engine of the decentralized finance (DeFi) movement. Maker is unlocking the power of the blockchain to deliver on the promise of economic empowerment today.

For more information, visit the MakerDAO website.

APPENDIX

Dai Use-Case Benefits and Examples

The Maker Protocol can be used by anyone, anywhere, without any restrictions or personal-information requirements. Below are a few examples of how Dai is used around the world:

Dai Offers Financial Independence to All

According to the World Bank's Global Findex Database 2017, about 1.7 billion adults around the world are unbanked.[5] In the US alone, according to a 2017 survey by the FDIC, around 32 million American households are either unbanked or underbanked,[6] meaning that they either have no bank account at all or they regularly use alternatives to traditional banking (e.g., payday or pawn shop loans) to manage their finances. Dai can empower every one of those people; all they need is access to the internet.

As the world's first unbiased stablecoin, Dai allows anyone to achieve financial independence, regardless of their location or circumstances. For example, in Latin America, Dai has provided an opportunity for individuals and families to hedge against the devaluation of the Argentine peso[7] and the Venezuelan bolívar. On the islands of Vanuatu in the South Pacific, where residents pay very high money transfer fees, Oxfam International, a U.K.-based non-profit; Australian startup, Sempo; and Ethereum startup

[5] https://globalfindex.worldbank.org/
[6]https://www.fdic.gov/householdsurvey/
[7] https://slideslive.com/38920018/living-on-defi-how-i-survive-argentinas-50-inflation

ConsenSys have successfully piloted a cash-assistance program through which 200 residents on the island of Efate were each given 50 Dai to pay a local network of vendors.[8]

Self-Sovereign Money Generation

Oasis Borrow allows users to access the Maker Protocol and generate Dai by locking their collateral in a Maker Vault. Notably, users do not need to access any third-party intermediary to generate Dai. Vaults offer individuals and businesses opportunities to create liquidity on their assets simply, quickly, and at relatively low cost.

Savings Earned Automatically

Dai holders everywhere can better power their journeys to financial inclusion by taking advantage of the Dai Savings Rate, which, as detailed earlier, builds on the value of Dai by allowing users to earn on the Dai they hold and protect their savings from inflation.

For example, if Bob has100,000 Dai locked in the DSR contract, and the DSR set by Maker Governance is 6% per year, Bob will earn savings of 6,000 Dai over 12 months. Additionally, because exchanges and blockchain projects can integrate the DSR into their own platforms, it presents new opportunities for cryptocurrency traders, entrepreneurs, and established businesses to increase their Dai savings and Dai operating capital. Due to this attractive mechanism, Market Makers, for example, may choose to hold their idle inventory in Dai and lock it in the DSR.

Fast, Low-cost Remittances

Cross-border remittances, whether for the purchase of goods or services or to simply send money to family and friends, can mean high service and transfer fees, long delivery timelines, and frustrating exchange issues due to inflation. The Dai stablecoin is used around the world as a medium of exchange because people have confidence in its value and efficiency.

Remittance users benefit from Dai in the following ways:

[8] https://www.coindesk.com/oxfam-trials-delivery-of-disaster-relief-using-ethereum-stablecoin-dai

- **Low-cost domestic and international transfers.** Dai provides immediate cost savings, as low gas fees replace high bank and wire service fees. Low cost allows for more frequent transactions.
- **Anytime service.** Dai doesn't rely on bank-like hours of operation. The Maker Protocol can be accessed 24/7/365.
- **Convenient on/off ramps.** Users can take advantage of the many fiat on and off ramps that exchange fiat currencies to Dai. These options allow users to bridge the gap between the fiat and cryptocurrency world, and easily cash out Dai holdings in their local currencies.
- **Increased security and confidence.** The blockchain offers high levels of security and consumer trust.

Stability in Volatile Markets

As noted above, Dai is both a readily accessible store of value and a powerful medium of exchange. As such, it can help protect traders from volatility. For example, it provides traders with a simple way to maneuver between positions smoothly and remain active in the market without having to cash out and repeat an on-ramp/off-ramp cycle.

Dai as an Ecosystem Driver and DeFi Builder

As more and more users become aware of Dai's value as a stablecoin, more developers are integrating it into the dapps they build on the Ethereum blockchain. As such, Dai is helping to power a more robust ecosystem. In short, Dai allows dapp developers to offer a stable method of exchange to their users who would rather not buy and sell goods and services using speculative assets.

Additionally, because Dai can be used to pay for gas in the Ethereum ecosystem, by creating DeFi dapps that accept Dai instead of ETH, developers offer users a smoother onboarding experience and a better overall experience.

Glossaries

- MakerDAO Glossary of Terms

- Maker Protocol Glossary (terms, variables, functions, and more)

System and Community Resources

- MakerDAO on GitHub
- MakerDAO Documentation
- MakerDAO.com
- The MakerDAO Blog
- The MakerDAO Forum
- The MakerDAO Chat
- MakerDAO on Reddit
- MakerDAO on Twitter

Compound: The Money Market Protocol

Version 1.0
February 2019

Authors
Robert Leshner, Geoffrey Hayes
https://compound.finance

Abstract
In this paper we introduce a decentralized protocol which establishes money markets with algorithmically set interest rates based on supply and demand, allowing users to frictionlessly exchange the time value of Ethereum assets.

Contents

1 Introduction

The market for cryptocurrencies and digital blockchain assets has developed into a vibrant ecosystem of investors, speculators, and traders, exchanging thousands [1] of blockchain assets. Unfortunately, the sophistication of financial markets hasn't followed: participants have little capability of trading the *time value* of assets.

Interest rates fill the gap between people with surplus assets they can't use, and people without assets (that have a productive or investment use); trading the time value of assets benefits both parties, and creates non-zero-sum wealth. For blockchain assets, two major flaws exist today:

- Borrowing mechanisms are extremely limited, which contributes to mispriced assets (e.g. "scamcoins" with unfathomable valuations, because there's no way to short them).
- Blockchain assets have negative yield, resulting from significant storage costs and risks (both on-exchange and off-exchange), without natural interest rates to offset those costs. This contributes to volatility, as holding is disincentivized.

Centralized exchanges (including Bitfinex, Poloniex...) allow customers to trade blockchain assets on margin, with "borrowing markets" built into the exchange. These are trust-based systems (you have to trust that the exchange won't get hacked, abscond with your assets, or incorrectly close out your position), are limited to certain customer groups, and limited to a small number of (the most mainstream) assets. Finally, balances and positions are virtual; you can't move a position on-chain, for example to use borrowed Ether or tokens in a smart contract or ICO, making these facilities inaccessible to dApps [2].

Peer to peer protocols facilitate collateralized and uncollateralized loans between market participants directly. Unfortunately, decentralization forces significant costs and frictions onto users; in every protocol reviewed, lenders are required to post, manage, and (in the event of collateralized loans) supervise loan offers and active loans, and loan fulfillment is often slow & asynchronous (loans have to be funded, which takes time) [3-6].

In this paper, we introduce a decentralized system for the frictionless borrowing of Ethereum tokens without the flaws of existing approaches, enabling proper money markets to function, and creating a safe positive-yield approach to storing assets.

2 The Compound Protocol

Compound is a protocol on the Ethereum blockchain that establishes money markets, which are pools of assets with algorithmically derived interest rates, based on the supply and demand for the asset. Suppliers (and borrowers) of an asset interact directly with the protocol, earning (and paying) a floating interest rate, without having to negotiate terms such as maturity, interest rate, or collateral with a peer or counterparty.

Each money market is unique to an Ethereum asset (such as Ether, an ERC-20 stablecoin such as Dai, or an ERC-20 utility token such as Augur), and contains a transparent and publicly-inspectable ledger, with a record of all transactions and historical interest rates.

2.1 Supplying Assets

Unlike an exchange or peer-to-peer platform, where a user's assets are matched and lent to another user, the Compound protocol aggregates the supply of each user; when a user supplies an asset, it becomes a fungible resource. This approach offers significantly more liquidity than direct lending; unless *every* asset in a market is borrowed (see below: the protocol incentivizes liquidity), users can withdraw their assets at any time, without waiting for a specific loan to mature.

Assets supplied to a market are represented by an ERC-20 token balance ("cToken"), which entitles the owner to an increasing quantity of the underlying asset. As the money market accrues interest, which is a function of borrowing demand, cTokens become convertible into an increasing amount of the underlying asset. In this way, earning interest is as simple as holding a ERC-20 cToken.

2.1.1 Primary Use Cases

Individuals with long-term investments in Ether and tokens ("HODLers") can use a Compound money market as a source of additional returns on their investment. For example, a user that owns Augur can supply their tokens to the Compound protocol, and earn interest (denominated in Augur) without having to manage their asset, fulfill loan requests or take speculative risks.

dApps, machines, and exchanges with token balances can use the Compound protocol as a source of monetization and incremental returns by "sweeping" balances; this has the potential to unlock entirely new business models for the Ethereum ecosystem.

2.2 Borrowing Assets

Compound allows users to frictionlessly borrow from the protocol, using cTokens as collateral, for use anywhere in the Ethereum ecosystem. Unlike peer-to-peer protocols, borrowing from Compound simply requires a user to specify a desired asset; there are no terms to negotiate, maturity dates, or funding periods; borrowing is instant and predictable. Similar to supplying an asset, each money market has a floating interest rate, set by market forces, which determines the borrowing cost for each asset.

2.2.1 Collateral Value

Assets held by the protocol (represented by ownership of a cToken) are used as collateral to borrow from the protocol. Each market has a collateral factor, ranging from 0 to 1, that represents the portion of the underlying asset value that can be borrowed. Illiquid, small-cap assets have low collateral factors; they do not make good collateral, while liquid, high-cap assets have high collateral factors. The sum of the value of an accounts underlying token balances, multiplied by the collateral factors, equals a user's ***borrowing capacity***.

Users are able to borrow up to, but not exceeding, their borrowing capacity, and an account can take no action (e.g. borrow, transfer cToken collateral, or redeem cToken collateral) that would raise the total value of borrowed assets above their borrowing capacity; this protects the protocol from default risk.

2.2.2 Risk & Liquidation

If the value of an account's borrowing outstanding exceeds their borrowing capacity, a portion of the outstanding borrowing may be repaid in exchange for the user's cToken collateral, at the current market price minus a ***liquidation discount***; this incentives an ecosystem of arbitrageurs to quickly step in to reduce the borrower's exposure, and eliminate the protocol's risk.

The proportion eligible to be closed, a ***close factor***, is the portion of the borrowed asset that can be repaid, and ranges from 0 to 1, such as 25%. The liquidation process may continue to be called until the user's borrowing is less than their borrowing capacity.

Any Ethereum address that possesses the borrowed asset may invoke the liquidation function, exchanging their asset for the borrower's cToken collateral. As both users, both assets, and prices are all contained within the Compound protocol, liquidation is frictionless and does not rely on any outside systems or order-books.

2.2.3 Primary Use Cases

The ability to seamlessly hold new assets (without selling or rearranging a portfolio) gives new superpowers to dApp consumers, traders and developers:

- Without having to wait for an order to fill, or requiring off-chain behavior, dApps can borrow tokens to use in the Ethereum ecosystem, such as to purchase computing power on the Golem network
- Traders can finance new ICO investments by borrowing Ether, using their existing portfolio as collateral
- Traders looking to short a token can borrow it, send it to an exchange and sell the token, profiting from declines in overvalued tokens

2.3 Interest Rate Model

Rather than individual suppliers or borrowers having to negotiate over terms and rates, the Compound protocol utilizes an interest rate model that achieves an interest rate equilibrium, in each money market, based on supply and demand. Following economic theory, interest rates (the "price" of money) should increase as a function of demand; when demand is low, interest rates should be low, and vise versa when demand is high. The utilization ratio ***U*** for each market ***a*** unifies supply and demand into a single variable:

$$U_a = Borrows_a \;/\; (Cash_a + Borrows_a)$$

The demand curve is codified through governance and is expressed as a function of utilization. As an example, borrowing interest rates may resemble the following:

$$Borrowing\ Interest\ Rate_a \ = \ 2.5\% + U_a * 20\%$$

The interest rate earned by suppliers is *implicit*, and is equal to the borrowing interest rate, multiplied by the utilization rate.

2.3.1 Liquidity Incentive Structure

The protocol does not guarantee liquidity; instead, it relies on the interest rate model to incentivize it. In periods of extreme demand for an asset, the liquidity of the protocol (the tokens available to withdraw or borrow) will decline; when this occur, interest rates rise, incentivizing supply, and disincentivizing borrowing.

3 Implementation & Architecture

At its core, a Compound money market is a ledger that allows Ethereum accounts to supply or borrow assets, while computing interest, a function of time. The protocol's smart contracts will be publicly accessible and completely free to use for machines, dApps and humans.

3.1 cToken Contracts

Each money market is structured as a smart contract that implements the ERC-20 token specification. User's balances are represented as cToken balances; users can `mint(uint amountUnderlying)` cTokens by supplying assets to the market, or `redeem(uint amount)` cTokens for the underlying asset. The price (exchange rate) between cTokens and the underlying asset increases over time, as interest is accrued by borrowers of the asset, and is equal to:

$$exchangeRate \ = \ \frac{underlyingBalance + totalBorrowBalance_a - reserves_a}{cTokenSupply_a}$$

As the market's total borrowing balance increases (as a function of borrower interest accruing), the exchange rate between cTokens and the underlying asset increases.

Function ABI	Description
`mint(uint256 amountUnderlying)`	Transfers an underlying asset into the market, updates msg.sender's cToken balance.

`redeem(uint256 amount)` `redeemUnderlying(uint256 amountUnderlying)`	Transfers an underlying asset out of the market, updates msg.sender's cToken balance.
`borrow(uint amount)`	Checks msg.sender collateral value, and if sufficient, transfers the underlying asset out of the market to msg.sender, and updates msg.sender's borrow balance.
`repayBorrow(uint amount)` `repayBorrowBehalf(address account, uint amount)`	Transfers the underlying asset into the market, updates the borrower's borrow balance.
`liquidate(address borrower, address collateralAsset, uint closeAmount)`	Transfers the underlying asset into the market, updates the borrower's borrow balance, then transfers cToken collateral from the borrower to msg.sender

Table 2. ABI and summary of primary cToken smart contract functions

3.2 Interest Rate Mechanics

Compound money markets are defined by an interest rate, applied to all borrowers uniformly, which adjust over time as the relationship between supply and demand changes.

The history of each interest rate, for each money market, is captured by an *Interest Rate Index*, which is calculated each time an interest rate changes, resulting from a user minting, redeeming, borrowing, repaying or liquidating the asset.

3.2.1 Market Dynamics

Each time a transaction occurs, the Interest Rate Index for the asset is updated to compound the interest since the prior index, using the interest for the period, denominated by r * t, calculated using a per-block interest rate:

$$Index_{a,n} = Index_{a,(n-1)} * (1 + r * t)$$

The market's total borrowing outstanding is updated to include interest accrued since the last index:

$$totalBorrowBalance_{a,n} = totalBorrowBalance_{\ a,(n-1)} * (1 + r * t)$$

And a portion of the accrued interest is retained (set aside) as reserves, determined by a ***reserveFactor***, ranging from 0 to 1:

$$reserves_a = reserves_{\ a,(n-1)} + totalBorrowBalance_{a,(n-1)} * (r * t * reserveFactor)$$

3.2.2 Borrower Dynamics

A borrower's balance, including accrued interest, is simply the ratio of the current index divided by the index when the user's balance was last checkpointed.

The balance for each borrower address in the cToken is stored as an *account checkpoint*. An account checkpoint is a Solidity tuple <*uint256 balance, uint256 interestIndex*>. This tuple describes the balance at the time interest was last applied to that account.

3.3 Borrowing

A user who wishes to borrow and who has sufficient balances stored in Compound may call `borrow(uint amount)` on the relevant cToken contract. This function call checks the user's account value, and given sufficient collateral, will update the user's borrow balance, transfer the tokens to the user's Ethereum address, and update the money market's floating interest rate.

Borrows accrue interest in the exact same fashion as balance interest was calculated in section 3.2; a borrower has the right to repay an outstanding loan at any time, by calling `repayBorrow(uint amount)` which repays the outstanding balance.

3.4 Liquidation

If a user's borrowing balance exceeds their total collateral value (borrowing capacity) due to the value of collateral falling, or borrowed assets increasing in value, the public function `liquidate(address target, address collateralAsset, address borrowAsset, uint closeAmount)` can be called, which exchanges the invoking user's asset for the borrower's collateral, at a slightly better than market price.

3.5 Price Feeds

A *Price Oracle* maintains the current exchange rate of each supported asset; the Compound protocol delegates the ability to set the value of assets to a committee which pools prices from the top 10 exchanges. These exchange rates are used to determine borrowing capacity and collateral requirements, and for all functions which require calculating the value equivalent of an account.

3.6 Comptroller

The Compound protocol does not support specific tokens by default; instead, markets must be whitelisted. This is accomplished with an admin function, `supportMarket(address market, address interest rate model)` that allows users to begin interacting with the asset. In order to borrow an asset, there must be a valid price from the Price Oracle; in order to use an asset as collateral, there must be a valid price and a collateralFactor.

Each function call is validated through a policy layer, referred to as the *Comptroller*; this contract validates collateral and liquidity, before allowing a user action to proceed.

3.7 Governance

Compound will begin with centralized control of the protocol (such as choosing the interest rate model per asset), and over time, will transition to complete community and stakeholder control. The following rights in the protocol are controlled by the admin:

- The ability to list a new cToken market
- The ability to update the interest rate model per market
- The ability to update the oracle address
- The ability to withdraw the reserve of a cToken
- The ability to choose a new admin, such as a DAO controlled by the community; because this DAO can itself choose a new admin, the administration has the ability to evolve over time, based on the decisions of the stakeholders

4 Summary

- Compound creates properly functioning money markets for Ethereum assets
- Each money market has interest rates that are determined by the supply and demand of the underlying asset; when demand to borrow an asset grows, or when supply is removed, interest rates increase, incentivizing additional liquidity
- Users can supply tokens to a money market to earn interest, without trusting a central party
- Users can borrow a token (to use, sell, or re-lend) by using their balances in the protocol as collateral

---

References

[1] Cryptocurrency Market Capitalizations. https://coinmarketcap.com/
[2] Bitfixex Margin Funding Guide. https://support.bitfinex.com/
[3] ETHLend White Paper. https://github.com/ETHLend
[4] Ripio White Paper. https://ripiocredit.network/
[5] Lendroid White Paper. https://lendroid.com/
[6] dYdX White Paper. https://whitepaper.dydx.exchange/
[7] Fred Ehrsam: The Decentralized Business Model. https://blog.coinbase.com/

AAVE

Protocol Whitepaper V1.0

wow@aave.com

January 2020

Abstract

This document describes the definitions and theory behind the Aave Protocol explaining the different aspects of the implementation.

Contents

1 Introduction

The birth of the Aave Protocol marks Aave's shift from a decentralized P2P lending strategy (direct loan relationship between lenders and borrowers, like in ETHLend) to a pool-based strategy. Lenders provide liquidity by depositing cryptocurrencies in a pool contract. Simultaneously, in the same contract, the pooled funds can be borrowed by placing a collateral. Loans do not need to be individually matched, instead they rely on the pooled funds, as well as the amounts borrowed and their collateral. This enables instant loans with characteristics based on the state of the pool. A simplified scheme of the protocol is presented in figure 1 below.

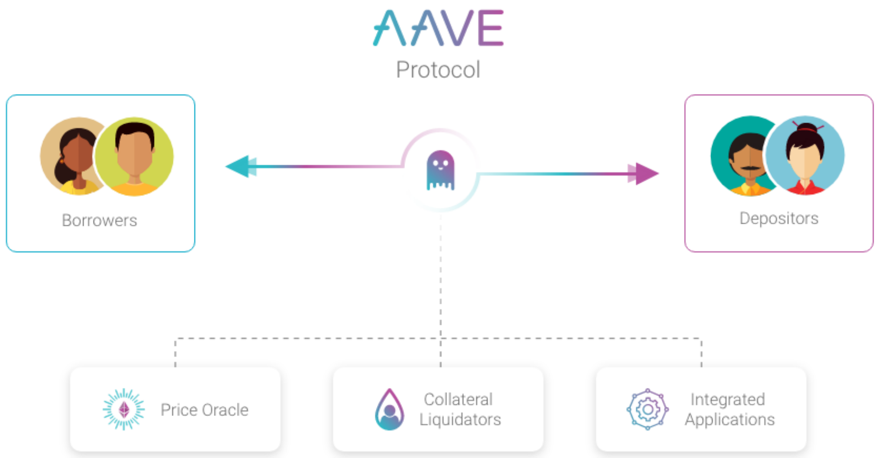


Figure 1: The Aave Protocol

The interest rate for both borrowers and lenders is decided algorithmically:

- For borrowers, it depends on the cost of money - the amount of funds available in the pool at a specific time. As funds are borrowed from the pool, the amount of funds available decreases which raises the interest rate.
- For lenders, this interest rate corresponds to the earn rate, with the algorithm safeguarding a liquidity reserve to guarantee withdrawals at any time.

1.1 Basic Concepts

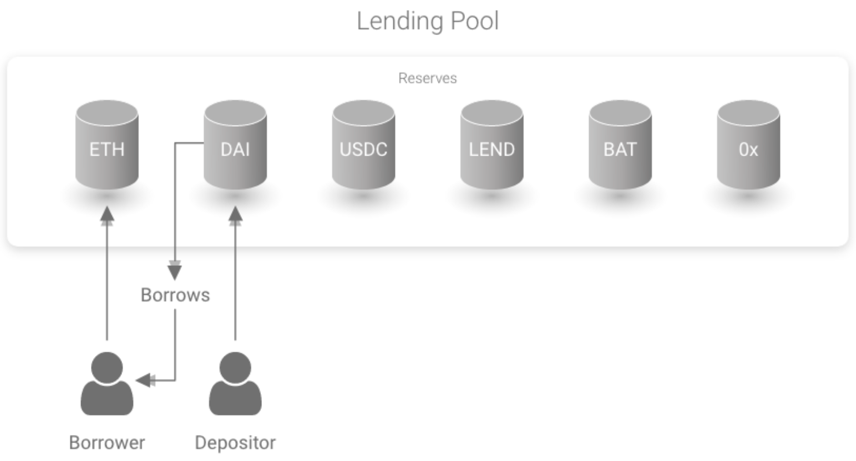


Figure 2: Lending Pool Basics

At the heart of a **lending pool** is the concept of **reserve**: every pool holds reserves in multiple currencies, with the total amount in Ethereum defined as **total liquidity**. A reserve accepts **deposits** from lenders. Users can

borrow these funds, granted that they lock a greater value as **collateral**, which backs the borrow position. Specific currencies in the pooled reserves can be configured as collateral or not for borrow positions, only low risk tokens should be considered. The amount one can borrow depends on the currencies deposited still available in the reserves. Every reserve has a specific **Loan-To-Value (LTV)**, calculated as the **weighted average** of the different LTVs of the currencies composing the collateral, where the weight for each LTV is the **equivalent amount of the collateral in ETH**; figure 3 shows an example of parameters.

Every borrow position can be opened with a **stable or variable rate**. Borrows have infinite duration, and there is no repayment schedule: partial or full repayments can be made anytime.

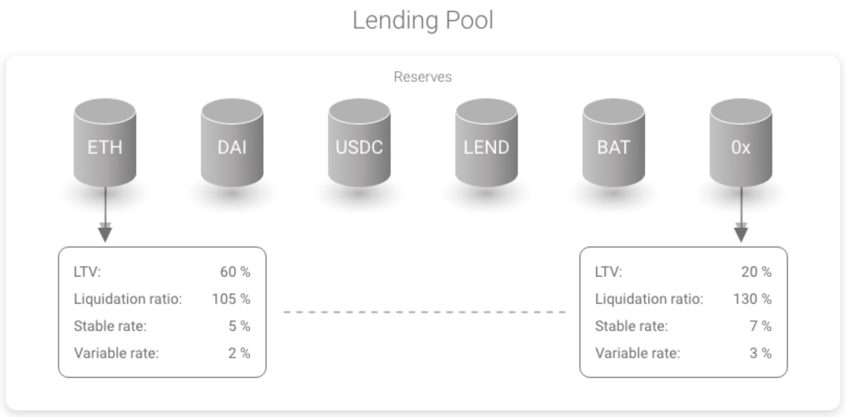


Figure 3: Lending Pool Parameters

In case of price fluctuations, a borrow position might be **liquidated**. A liquidation event happens when the price of the collateral drops below the threshold, L_Q, called **liquidation threshold**. Reaching this ratio channels a **liquidation bonus**, which incentivizes liquidators to buy the collateral at a discounted price. Every reserve has a specific liquidation threshold, following the same approach as for the LTV. Calculation of the average liquidation threshold L_Q^a is performed dynamically, using the weighted average of the liquidation thresholds of the collateral's underlying assets.

At any point in time, a borrow position is characterized by its health factor H_f, a function for the total collateral and the total borrows which determines if a loan is **undercollateralized**:

$$H_f = \frac{TotalCollateralETH * L_Q^a}{TotalBorrowsETH + TotalFeesETH} \; when \; H_f < 1, \; a \; loan \; is \; considered \; undercollateralized \; and \; can \; be \; liquidated$$

Further details on liquidation can be found in section 3.6.

1.2 Formal Definitions

Variable	Description	
T, **current timestamp**	Current number of seconds defined by `block.timestamp`.	
T_l, **last updated timestamp**	Timestamp of the last update of the reserve data. T_l is updated every time a borrow, deposit, redeem, repay, swap or liquidation event occurs.	
ΔT, **delta time**		$\Delta T = T - T_l$
T_{year}, **seconds**	Number of seconds in a year.	$T_{year} = 31536000$
ΔT_{year}, **yearly period**		$\Delta T_{year} = \frac{\Delta T}{T_{year}}$
L_t, **total liquidity**	Total amount of liquidity available in the reserve. The decimals of this value depend on the decimals of the currency.	
B_s, **total stable borrows**	Total amount of liquidity borrowed at a stable rate. The decimals of this value depend on the decimals of the currency.	
B_v, **total variable borrows**	Total amount of liquidity borrowed at a variable rate. The decimals of this value depend on the decimals of the currency.	
B_t, **total borrows**	Total amount of liquidity borrowed. The decimals of this value depend on the decimals of the currency.	$B_t = B_s + B_v$
U, **utilization rate**	Representing the utilization of the deposited funds.	$U = \begin{cases} 0, \text{ if } L_t = 0 \\ \frac{B_t}{L_t}, \text{ if } L_t > 0 \end{cases}$
$U_{optimal}$, **target utilization rate**	The utilization rate targeted by the model, beyond the variable interest rate rises sharply.	
R_{v_0}, **base variable borrow rate**	Constant for $B_t = 0$. Expressed in ray.	
R_{slope1}, **interest rate slope below** $U_{optimal}$	Constant representing the scaling of the interest rate versus the utilization, when $U < U_{optimal}$. Expressed in ray.	
R_{slope2}, **interest rate slope above** $U_{optimal}$	Constant representing the scaling of the interest rate versus the utilization, when $U \geq U_{optimal}$. Expressed in ray.	
R_v, **variable borrow rate**	$R_v = \begin{cases} R_{v_0} + \frac{U}{U_{optimal}} R_{slope1}, \text{ if } U \leq U_{optimal} \\ R_{v_0} + R_{slope1} + \frac{U - U_{optimal}}{1 - U_{optimal}} R_{slope2}, \text{ if } U > U_{optimal} \end{cases}$	
R_s, **stable rate**	Implemented in section 4.2. Expressed in ray.	
M_r, **average market lending rate**	Base stable borrow rate, defined for i platforms with P_i^r the lending rate and P_i^v the borrowing volume. Expressed in ray.	$M_r = \frac{\sum_{i=1}^{n} P_r^i P_v^i}{\sum_{i=1}^{n} P_v^i}$

Variable	Description	
R^t_{sa}, **average stable rate borrow rate**	When a stable borrow of amount B_{new} is issued at rate R_s:	$\mathrm{R}^t_{sa} = \frac{B_s R^{t-1}_{sa} + B_{new} R_s}{B_s + B_{new}}$
	When a user repays an amount B_x at stable rate R_{sx}:	$R^t_{sa} = \begin{cases} 0, \text{ if } B_s - B_x = 0 \\ \frac{B_s R^{t-1}_{sa} - B_x R_{sx}}{B_s - B_x}, \text{ if } B_s - B_x > 0 \end{cases}$
	Check the methods `decreaseTotalBorrowsStableAndUpdateAverageRate()` and `increaseTotalBorrowsStableAndUpdateAverageRate()`. Expressed in ray.	
R_O, **overall borrow rate**	Overall borrow rate of the reserve, calculated as the weighted average between the total variable borrows B_v and the total stable borrows B_s.	$R_O = \begin{cases} 0, \text{ if } B_t = 0 \\ \frac{B_v R_v + B_s R_{sa}}{B_t}, \text{ if } B_t > 0 \end{cases}$
R_l, **current liquidity rate**	Function of the overall borrow rate R_O and the utilization rate U.	$R_l = R_O U$
C^t_i, **cumulated liquidity index**	Interest cumulated by the reserve during the time interval Δ_T, updated whenever a borrow, deposit, repay, redeem, swap, liquidation event occurs.	$C^t_i = (R_l \Delta T_{year} + 1) C^{t-1}_i$ $C^0_i = 1 \times 10^{27} = 1\ ray$
I^t_n, **reserve normalized income**	Ongoing interest cumulated by the reserve.	$I^t_n = (R_l \Delta T_{year} + 1) C^{t-1}_i$
B^t_{vc}, **cumulated variable borrow index**	Interest cumulated by the variable borrows B_v, at rate R_v, updated whenever a borrow, deposit, repay, redeem, swap, liquidation event occurs.	$B^t_{vc} = (1 + \frac{R_v}{T_{year}})^{\Delta T_x} B^{t-1}_{vc}$ $B^0_{vc} = 1 \times 10^{27} = 1\ ray$
B^t_{vcx}, **user cumulated variable borrow index**	Variable borrow index of the specific user, stored when a user opens a variable borrow position.	$B^t_{vcx} = B^t_{vc}$
B_x, **user principal borrow balance**	Balance stored when a user opens a borrow position. In case of multiple borrows, the compounded interest is cumulated each time and it becomes the new principal borrow balance.	
B_{xc}, **user compounded borrow balance**	Principal B_x plus the cumulated interests.	
	For a variable position:	$B_{xc} = \frac{B_{vc}}{B_{vcx}} (1 + \frac{R_v}{T_{year}})^{\Delta T_x} B_x$
	For a stable position:	$B_{xc} = (1 + \frac{R_s}{T_{year}})^{\Delta T_x} B_x$
H_f, **health factor**	when $H_f < 1$, a loan is considered undercollateralized and can be liquidated	$H_f = \frac{TotalCollateralETH * L^a_Q}{B_t + TotalFeesETH}$

2 Protocol Architecture

The current implementation of the protocol is as follows:

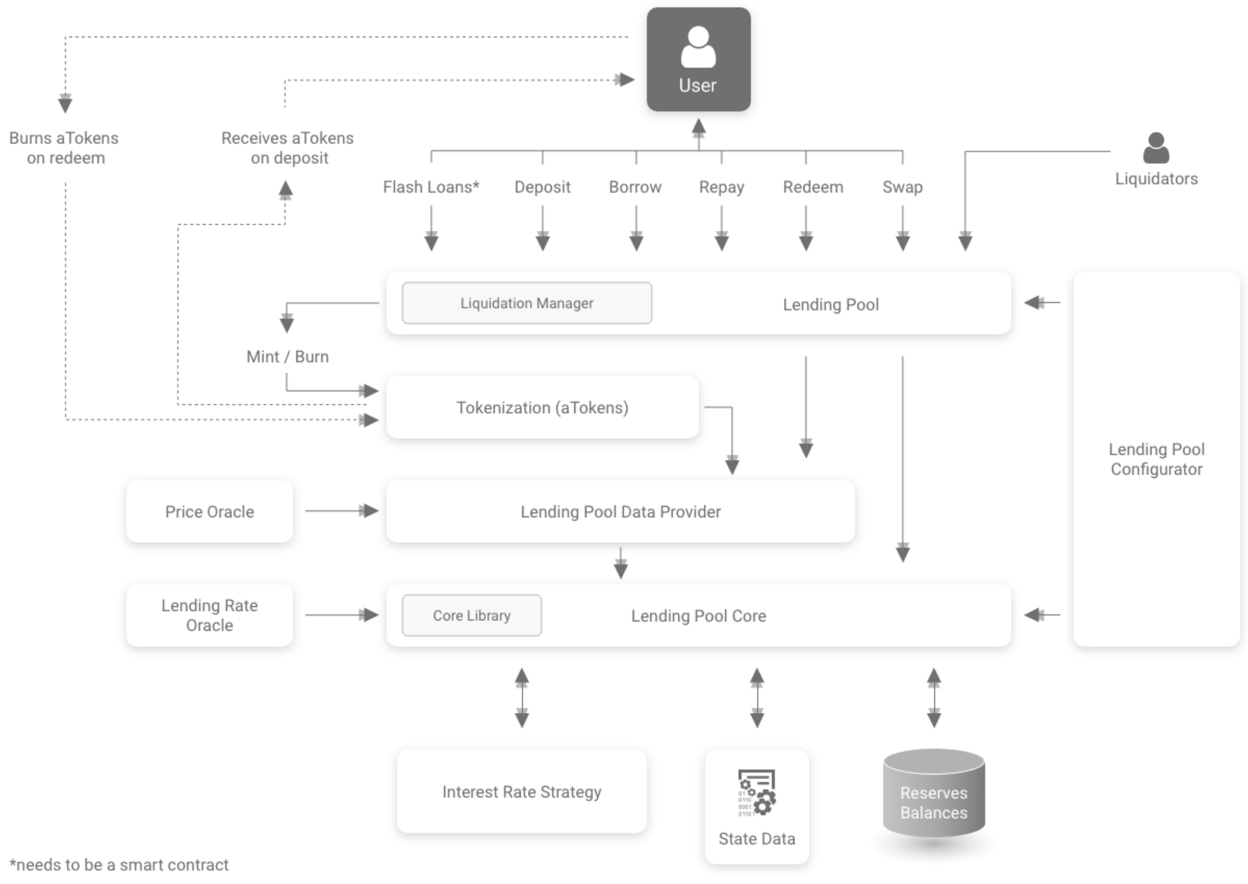


Figure 4: Protocol Architecture

2.1 Lending Pool Core

The `LendingPoolCore` contract is the center of the protocol, it:

- holds the state of every reserve and all the assets deposited,
- handles the basic logic (cumulation of the indexes, calculation of the interest rates...).

2.2 Lending Pool Data Provider

The `LendingPoolDataProvider` contract performs calculations on a higher layer of abstraction than the `LendingPoolCore` and provides data for the `LendingPool`; specifically:

- Calculates the ETH equivalent a user's balances (Borrow Balance, Collateral Balance, Liquidity Balance) to assess how much a user is allowed to borrow and the health factor.
- Aggregates data from the `LendingPoolCore` to provide high level information to the `LendingPool`.
- Calculate of the Average Loan to Value and Average Liquidation Ratio.

2.3 Lending Pool

The `LendingPool` contract uses the `LendingPoolCore` and `LendingPoolDataProvider` to interact with the reserves through the actions:

- Deposit
- Redeem
- Borrow
- Repay
- Rate swap
- Liquidation
- Flash loan

One of the advanced features implemented in the `LendingPool` contract is the **tokenization** of the lending position. When a user deposits in a specific reserve, he receives a corresponding amount of **aTokens**, tokens that map the liquidity deposited and accrue the interests of the deposited underlying assets. Atokens are minted upon deposit, their value increases until they are burned on redeem or liquidated. Whenever a user opens a borrow position, the tokens used as collateral are locked and cannot be transferred. Further details on the tokenization are in section 3.8.

2.4 Lending Pool Configurator

The `LendingPoolConfigurator` provides main configuration functions for `LendingPool` and `LendingPoolCore`:

- Reserve initialization
- Reserve configuration
- Enable/disable borrowing on a reserve
- Enable/disable the usage of a specific reserve as collateral.

The `LendingPoolConfigurator` contract will be integrated in Aave Protocol governance.

2.5 Interest Rate Strategy

The `InterestRateStrategy` contract holds the information needed to update the interest rates of a specific reserve and implements the update of the interest rates. Every reserve has a specific `InterestRateStrategy` contract. Specifically, within the base strategy contract `DefaultReserveInterestRateStrategy` the following are defined:

- Base variable borrow rate R_{v_0}
- Interest rate slope below optimal utilisation R_{slope1}
- Interest rate slope beyond optimal utilisation R_{slope2}

The current variable borrow rate is:

$$R_v = \begin{cases} R_{v_0} + \frac{U}{U_{optimal}} R_{slope1}, \text{ if } U \leq U_{optimal} \\ R_{v_0} + R_{slope1} + \frac{U - U_{optimal}}{1 - U_{optimal}} R_{slope2}, \text{ if } U > U_{optimal} \end{cases}$$

This interest rate model allows for calibration of key interest rates:

- At $U = 0$, $R_v = R_{v_0}$
- At $U = U_{optimal}$, $R_v = R_{v_0} + R_{slope1}$
- Above $U_{optimal}$, the interest rate rises sharply to take into account the cost of capital.

The stable borrow rate follows the same model described in section 4.2.

2.6 Governance

The rights of the protocol are controlled by the LEND token. Initially, the Aave Protocol will be launched with a decentralized on-chain governance based on the DAOStack framework which will evolve to a fully autonomous protocol. On-chain implies all votes are binding: actions that follow a vote are hard-coded and must be executed.

To understand the scope of the governance it's important to make the distinction:

- The **Aave Protocol** is bound to evolve and will allow the creation of multiple lending pools with segregated liquidity, parameters, permissions, and type of assets.
- The **Aave Lending Pool** is the first pool of the Aave protocol until the Pool Factory Update is released and anyone can create their own pool.

Within the Aave Protocol, the governance will take place at two level :

1. The **Protocol's Governance** voting is weighted by LEND for decisions related to protocol parameters and upgrades of the smart contract. It can be compared to MakerDAO's governance where stakeholders vote on current and future parameters of the protocol.
2. The **Pool's Governance** where your vote is weighted based on your share of pool liquidity expressed in aTokens. The votes cover pool specific parameters such as assets used as collateral or to be borrowed.

   Each Pool will have its own governance, under the umbrella of the Protocol's Governance.

More details on the Governance will be published in a Governance Proposal to the community.

3 The `LendingPool` Contract

The actions implemented within `LendingPool` allow users to interact with the reserve. All the actions follow this specific sequence:

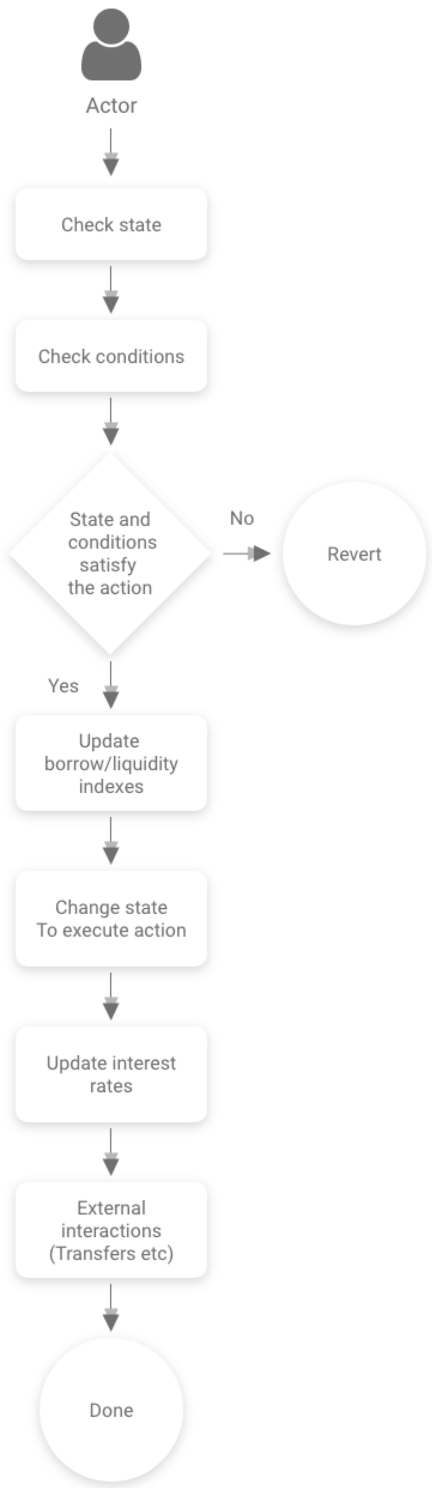

Actor
Check state
Check conditions
State and conditions satisfy the action
No
Revert
Yes
Update borrow/liquidity indexes
Change state To execute action
Update interest rates
External interactions (Transfers etc)
Done

Figure 5: The `LendingPool` Contract

3.1 Deposit

The deposit action is the simplest one and does not have any particular state check. The sequence of action is:

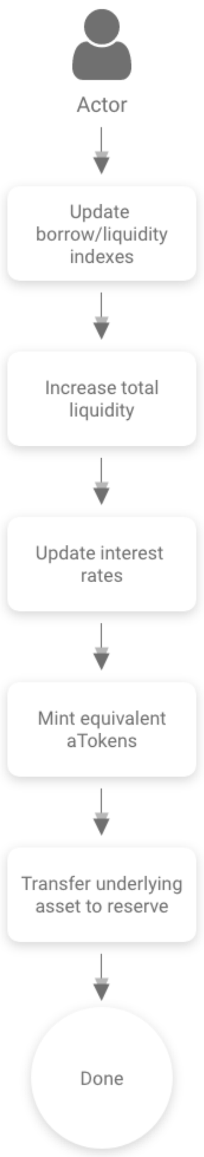


Figure 6: Deposit funds

3.2 Redeem

The redeem action allows users to exchange an amount of aTokens for the underlying asset. The actual amount to redeem is calculated using the aToken/underlying exchange rate E_i in section 3.8. The action is defined as follows:

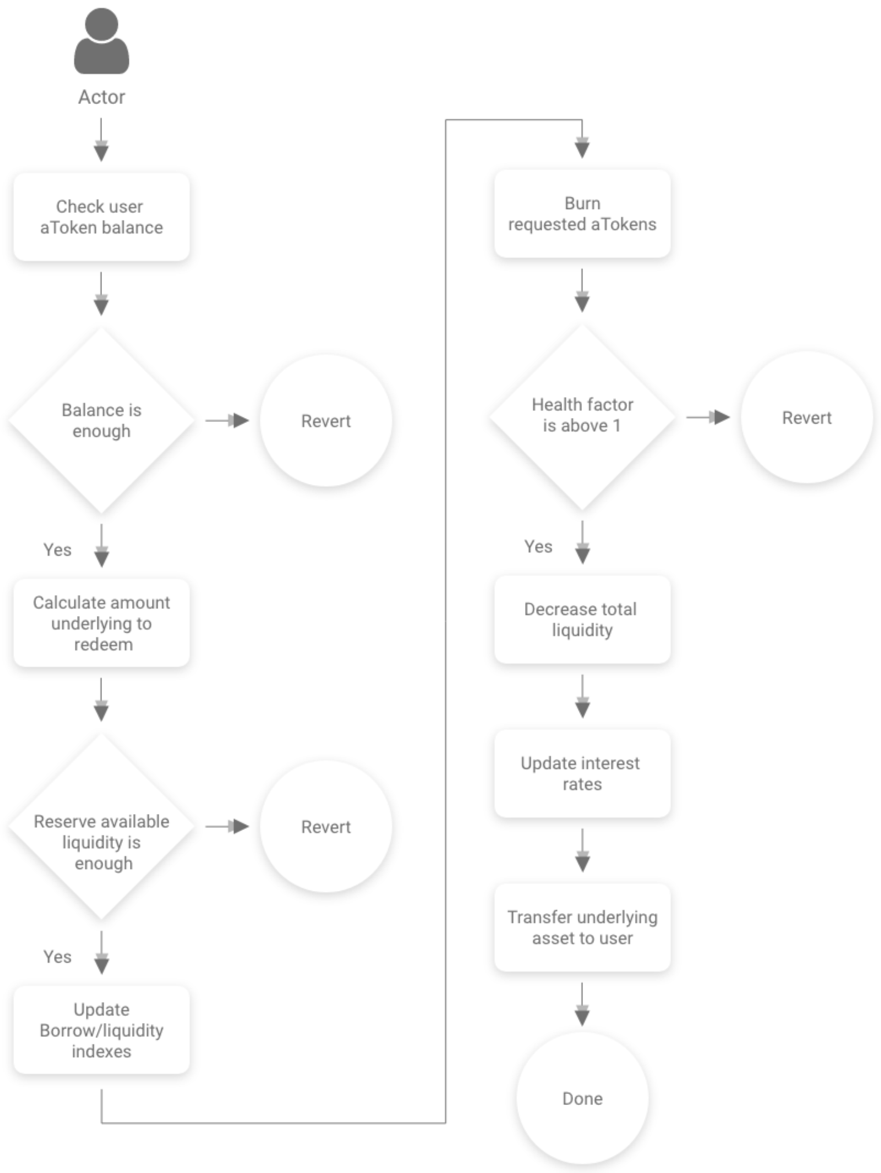

Actor
Check user aToken balance
Balance is enough
Revert
Yes
Calculate amount underlying to redeem
Reserve available liquidity is enough
Revert
Yes
Update Borrow/liquidity indexes
Burn requested aTokens
Health factor is above 1
Revert
Yes
Decrease total liquidity
Update interest rates
Transfer underlying asset to user
Done

Figure 7: Redeem funds

3.3 Borrow

The borrow action transfers to the user a specific amount of underlying asset, in exchange of a collateral that remains locked. The flow of action can be described as follows:

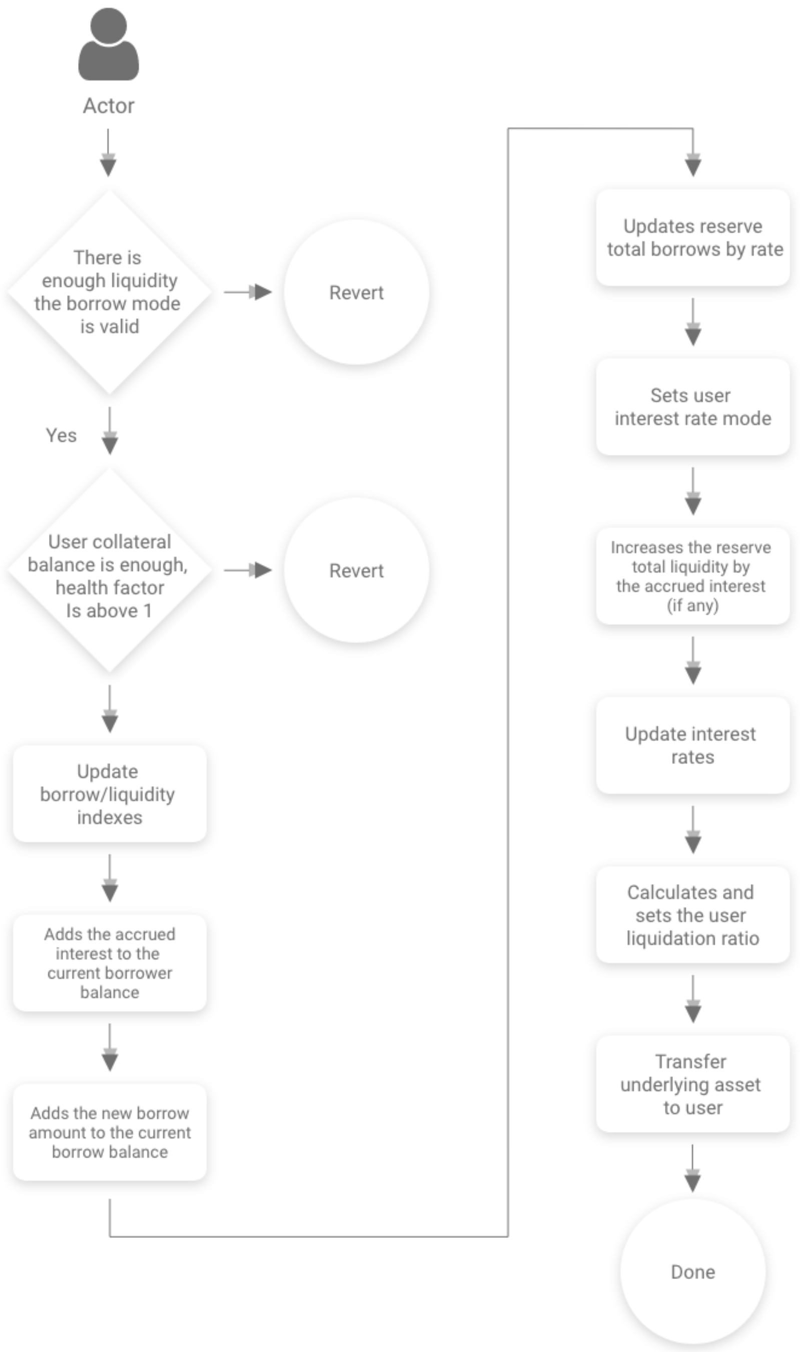
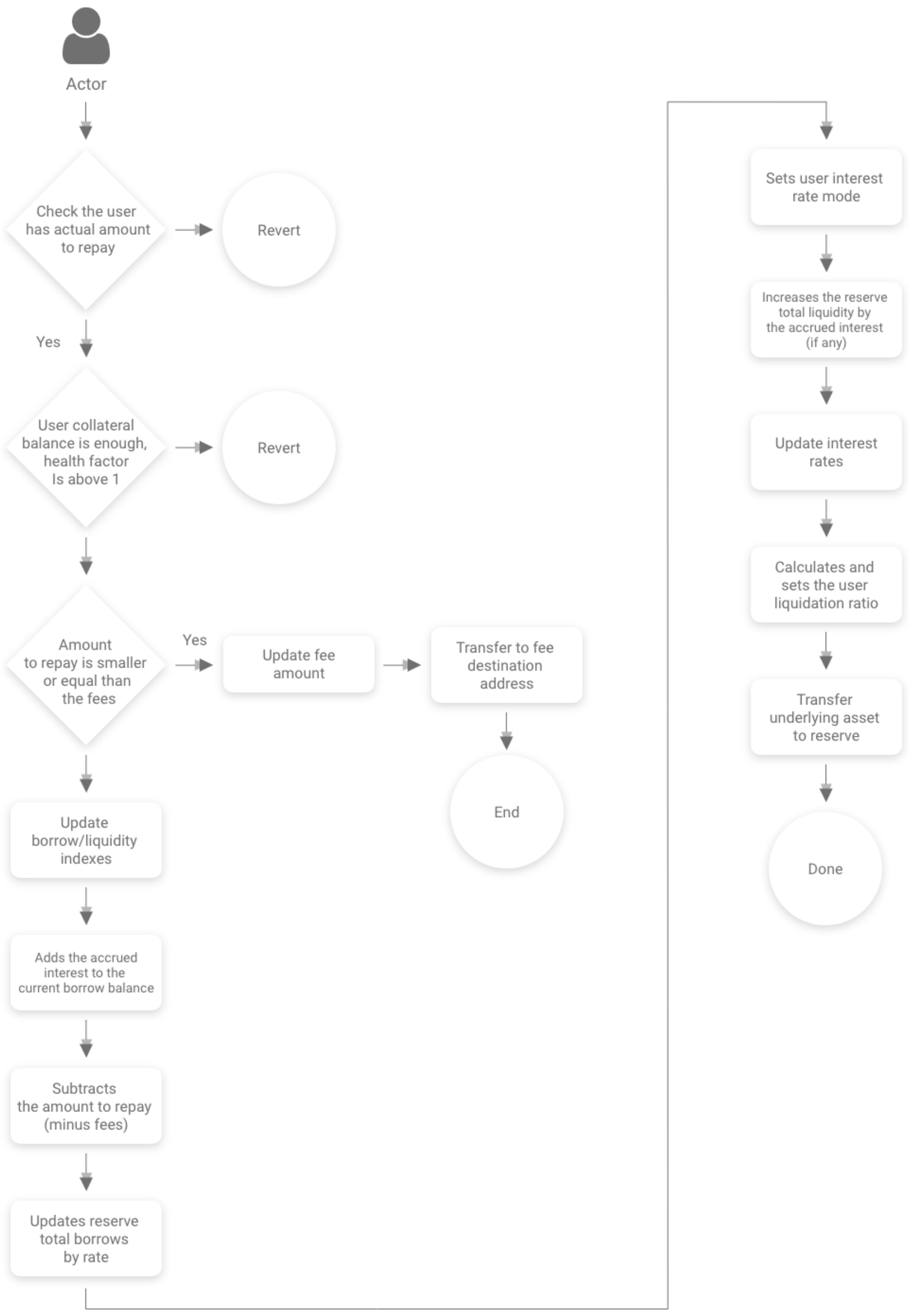

Actor
There is enough liquidity the borrow mode is valid
Revert
Yes
User collateral balance is enough, health factor Is above 1
Revert
Update borrow/liquidity indexes
Adds the accrued interest to the current borrower balance
Adds the new borrow amount to the current borrow balance
Updates reserve total borrows by rate
Sets user interest rate mode
Increases the reserve total liquidity by the accrued interest (if any)
Update interest rates
Calculates and sets the user liquidation ratio
Transfer underlying asset to user
Done

Figure 8: Borrow funds

3.4 Repay

The repay action allows the user to repay completely or partially the borrowed amount plus the origination fee and the accrued interest.



Actor
Check the user has actual amount to repay
Revert
Yes
User collateral balance is enough, health factor Is above 1
Revert
Amount to repay is smaller or equal than the fees
Yes
Update fee amount
Transfer to fee destination address
End
Update borrow/liquidity indexes
Adds the accrued interest to the current borrow balance
Subtracts the amount to repay (minus fees)
Updates reserve total borrows by rate
Sets user interest rate mode
Increases the reserve total liquidity by the accrued interest (if any)
Update interest rates
Calculates and sets the user liquidation ratio
Transfer underlying asset to reserve
Done

Figure 9: Repay a loan

3.5 Swap Rate

The swap rate action allows a user with a borrow in progress to swap between variable and stable borrow rate.

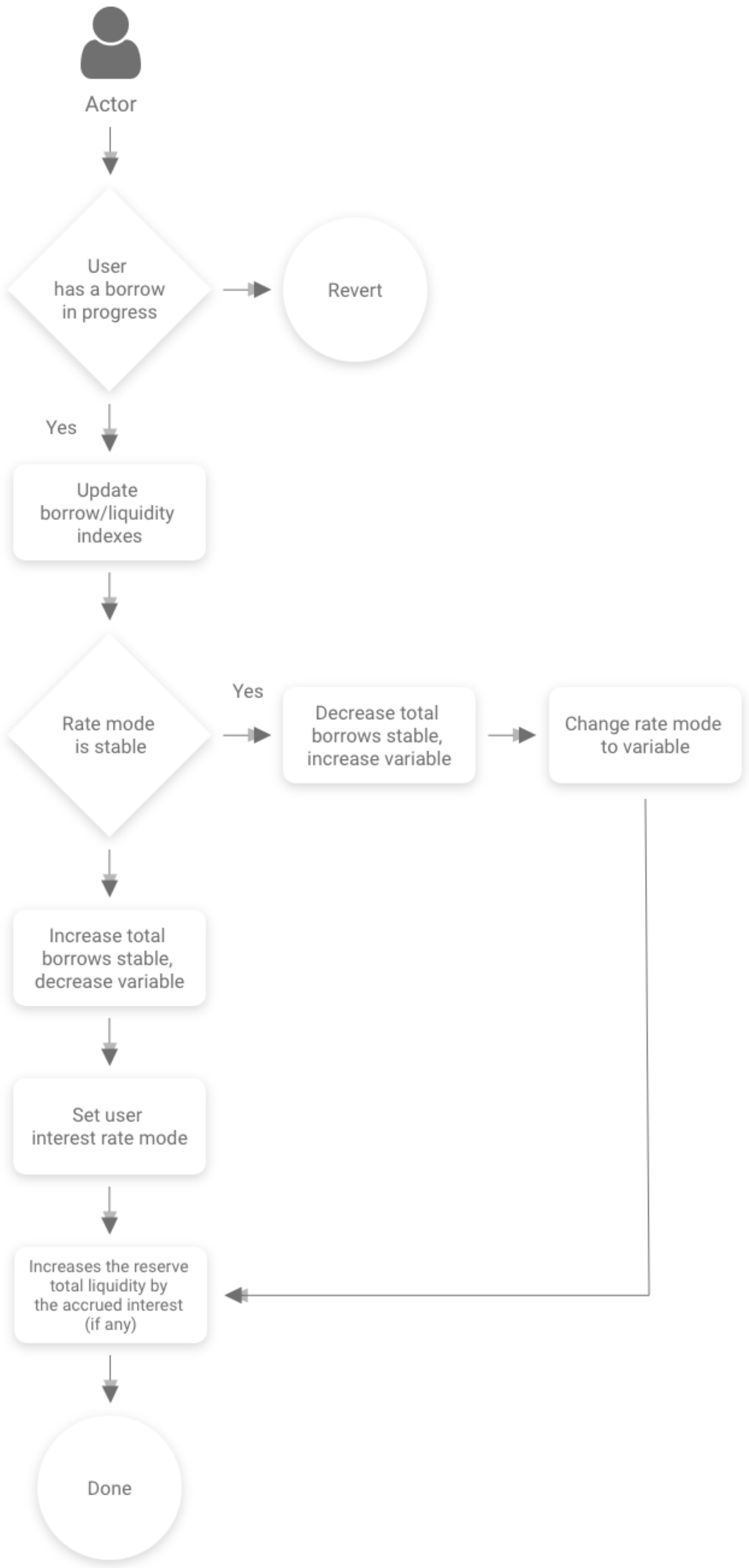

Actor
User has a borrow in progress
Revert
Yes
Update borrow/liquidity indexes
Rate mode is stable
Yes
Decrease total borrows stable, increase variable
Change rate mode to variable
Increase total borrows stable, decrease variable
Set user interest rate mode
Increases the reserve total liquidity by the accrued interest (if any)
Done

Figure 10: Swap Rate

3.6 Liquidation Call

The `liquidationcall` contract allows any external actor to purchase part of a collateral at a discounted price. In case of a liquidation event, a maximum of 50% of the loan can be liquidated, which will bring the health factor back above 1.

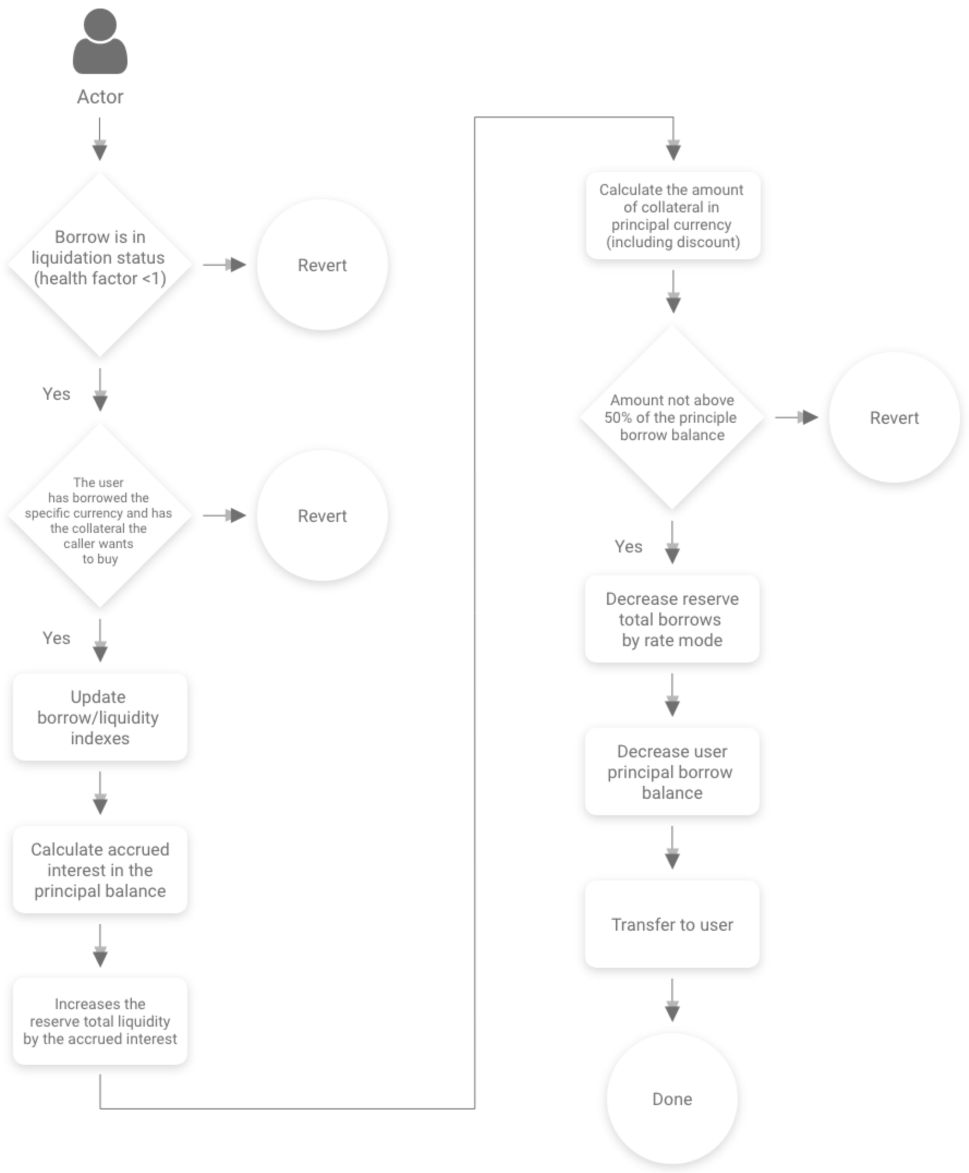

Actor
Borrow is in liquidation status (health factor <1)
Revert
Yes
The user has borrowed the specific currency and has the collateral the caller wants to buy
Revert
Yes
Update borrow/liquidity indexes
Calculate accrued interest in the principal balance
Increases the reserve total liquidity by the accrued interest
Calculate the amount of collateral in principal currency (including discount)
Amount not above 50% of the principle borrow balance
Revert
Yes
Decrease reserve total borrows by rate mode
Decrease user principal borrow balance
Transfer to user
Done

Figure 11: Liquidation

3.7 Flash Loans

The flash loan action will allow users to borrow from the reserves within a single transaction, as long as the user returns more liquidity that has been taken.

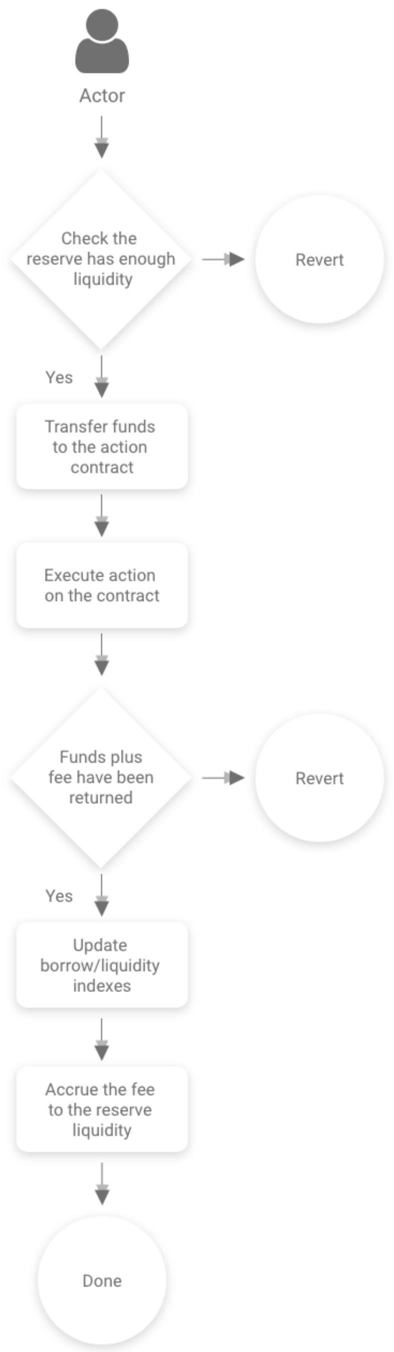

Actor
Check the reserve has enough liquidity
Revert
Yes
Transfer funds to the action contract
Execute action on the contract
Funds plus fee have been returned
Revert
Yes
Update borrow/liquidity indexes
Accrue the fee to the reserve liquidity
Done

Figure 12: Flash Loan

Flash loans temporarily transfer the funds to a smart contract that respects the `IFlashLoanEnabledContract.sol` interface. The address of the contract is a parameter of the action. After the funds are transferred, the method `executeOperation()` is executed on the external contract. The contract can do whatever action is needed with the borrowed funds. After the method `executeOperation()` is completed, a check is performed to verify that the funds plus fee have been returned to the `LendingPool` contract. The fee is then accrued to the reserve, and the state of the reserve is updated. If less funds than what was borrowed have been returned to the reserve, the transaction is reverted.

3.8 Tokenization

The Aave protocol implements a tokenization strategy for liquidity providers. Upon deposit, the depositor receives a corresponding amount of derivative tokens, called Aave Tokens (aTokens for short) that map 1:1 the underlying assets. The balance of aTokens of every depositor grows over time, driven by the perpetual accrual of interest of deposits. aTokens are fully ERC20 compliant.

aTokens also natively implement the concept of interest rate redirection. Indeed, the value accrued over time by the borrowers' interest rate payments is distinct from the principal value. Once there is a balance of aTokens, the accrued value can be redirected to any address, effectively splitting the balance and the generated interest. We call the continuous flow of accumulated interest over time the **interest stream**.

To implement this tokenization strategy, Aave introduced the following concepts in the aToken contract:

1. **User x balance index I_x^t:** Is the value of the reserve normalized income I_x^t at the moment of execution of the last action by the user.
2. **Principal balance B_p:** Is the balance stored in the balances mapping of the ERC20 aToken contract. The principal balance gets updated on every action that the user executes on the aToken contract (deposit, redeem, transfer, liquidation, interest rate redirection)
3. **Redirection address A_r:** When a user decides to redirect his interest stream to another address, a new redirection address A_r is provided. If no redirection of the interest stream is performed, A_r is 0
4. **Redirected Balance B_r^x:** Whenever a user redirects his interest stream, the balance of user redirecting is added to the redirected balance B_r of the address specified by B_r. Defined as follows:

   $$B_r^x = \sum_X B_p$$

   Where X is the set of users redirecting the interest stream to the user x

   The redirected balance decreases whenever a user $x_0 \in X$ redeems or transfers his aTokens to another user that is not redirecting to x.
5. **Current balance** B_c: Is the balance returned by the `balanceOf()` function of the aToken contract. Defined as follows:

   $$\mathrm{B}_c^x = \begin{cases} 0\text{, if } B_p^x = 0 \text{ and } B_r^x = 0 \\ B_p^x + B_r^x(\frac{I_n}{I_x} - 1)\text{, if } A_r <> 0 \\ B_p^x \frac{I_n}{I_x} + B_r^x(\frac{I_n}{I_x} - 1)\text{, if } A_r = 0 \end{cases}$$

3.8.1 Limitations of the tokenization model

The described tokenization model has many advantages compared to the widely used, exchange rate based approach, but also some drawbacks, specifically:

1. **It's impossible to transfer the whole balance at once:** Given the perpetual accrual of the interest rate, there is no way to specify the exact amount to transfer, since the interest will keep accruing even while the transfer transaction is being confirmed. This means that having exactly 0 balance after a transfer is impossible, rather, a very small balance (dust balance) will be left to the from account executing the transfer. Note that this could have been avoided by adding specific logic to handle this particular edge case, but this would have meant adding a non standard behavior to the ERC20 transfer function, and for this reason we avoided it. Even though this is not a relevant issue, it's important to note that is possible to completely clear the remaining balance by either 1. execute another transfer, which will most likely transfer the remaining dust balance as it would be too small to accrue interest in a reasonably short amount of time, or 2. redeem the dust balance and transfer the underlying asset.
2. **Interest stream can only be redirected if there is a principal balance:** This means that only accounts that have a principal balance B_p can redirect their interest. If users redeem or transfer everything, their interest redirection is reset. As a side effect of this, interest generated only by the redirected balance B_r cannot be redirected.

4 Stable Rate Theory

The following chapter explains how the stable rates are applied to the system and the limitations.

Implementation of a fixed rate model on top of a pool is complicated. Indeed, fixed rates are hard to handle algorithmically, as the cost of borrowing money varies with market conditions and the liquidity available. There might therefore be situations (sudden market changes, bank runs ...) in which handling stable rate borrow positions would need using specific heuristics based on time or economical constraints. Following this reasoning, we identified two possible ways of handling fixed rates:

1. **Imposing time constraints**: fixed rates might work perfectly fine in a time constrained fashion. If a loan has a stable duration, it should survive extreme market conditions, as the borrower must repay at the end of the loan period. Unfortunately, time constrained fixed rate loans aren't suitable for our specific use case of open ended loan. It would require a certain degree of UX friction where users would need to create and handle multiple loans with different times constraints.

2. **Imposing rates constraints**: An interest rate calculated at the beginning of a loan might be impacted by market conditions, keeping it from staying fixed. If the rate diverges too much from the market, it can be readjusted. This would not be a pure fixed rate, open term loan - as the rate might vary throughout the loan duration – yet users will experience actual fixed rates during specific time periods, or when there is enough liquidity available. This particular implementation has been chosen to be integrated into Aave's Protocol under the name **stable rate**.

4.1 Lending Rate Oracle

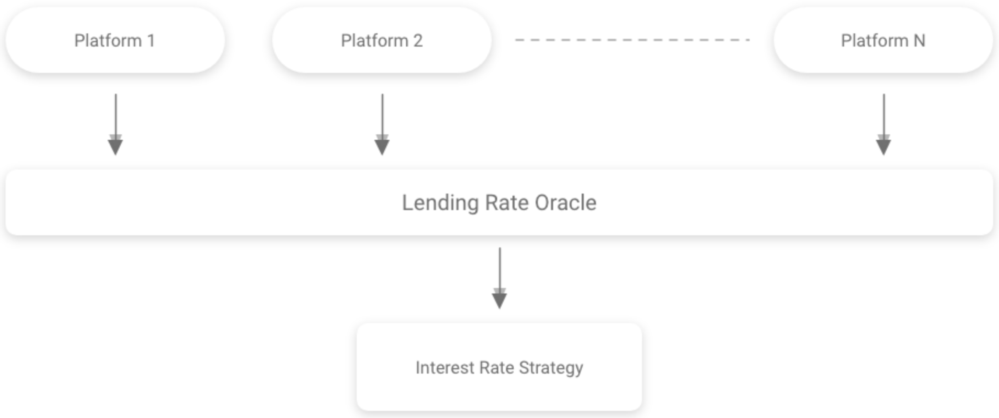


Figure 13: Lending Rate Oracle

The first component to be integrated into the Protocol protocol is a Lending Rate Oracle, which will provide information to the contracts on the actual market rates that other lending platforms, both centralized and decentralized, are providing. The average market lending rate M_r is defined for i platforms with P_r^i the lending rate and P_v^i the borrowing volume:

$$M_r = \frac{\sum_{i=1}^{n} P_r^i P_v^i}{\sum_{i=1}^{n} P_v^i}$$

The market rate will be updated daily, initially by Aave.

4.2 Current Stable Borrow Rate R_s

The current stable borrow rate is calculated as follows:

$$R_s^t = \begin{cases} M_r + \frac{U}{U_{optimal}} R_{slope1}, \text{ if } U \leq U_{optimal} \\ M_r + R_{slope1} + \frac{U - U_{optimal}}{1 - U_{optimal}} R_{slope2}, \text{ if } U > U_{optimal} \end{cases}$$

With:
- M_r the average market lending rate.
- R_{slope1} the interest rate slope below $U_{optimal}$, increases the rate as U increases.
- R_{slope2} the interest rate slope beyond $U_{optimal}$, increases as the difference between U and $U_{optimal}$ increases.
- U is the utilization rate.

Note: R_s does NOT impact existing stable rates positions – this is applied only to new opened positions.

4.3 Limitations on Stable Rate Positions

To avoid abuses on stable rate loans, the following limitations have been applied to the stable rate borrowing model:

1. Users cannot deposit as collateral more liquidity than what they are trying to borrow. Eg. a user deposits 10 million DAI collateral, tries to borrow 1 million DAI. This is to prevent the following attack vector:

   $$\text{Given: } B_s = 18\%APR,\ M_r = 9\%APR,\ R_l = 12\%APR$$

   Users might try to artificially lower B_s to the value of M_r by depositing a huge amount of liquidity which would cause B_s to drop, then borrow from the same liquidity at a lower rate, withdraw the liquidity previously deposited to cause B_s and the liquidity rate R_l to raise again; then finally deposit the amount borrowed to earn interest on the previously borrowed funds. Although this attack can still be carried out using multiple accounts, this particular constraint makes the attack more complicated as it requires more money (and a different collateral currency). This works well in combination with the interest rate rebalancing in the next section.

2. Borrowers will only be able to borrow up to T_r of the available liquidity at the current borrow rate. So, for every specific value of B_s, there is only up to T_r of liquidity available for a single borrower. This is to avoid that a specific borrower would borrow too much available liquidity at a too competitive rate.

4.4 Stable Rate Rebalancing

The last and perhaps most important constraint of the stable rate model is the rate rebalancing. This is to work around changes in market conditions or increased cost of money within the pool.
The stable rate rebalancing will happen in two specific situations:

1. **Rebalancing up**. The stable rate of a user x is rebalanced to the most recent value of B_s when a user could earn interest by borrowing:

   $$B_s^x < R_l \text{ with } B_s^x \text{ the stable borrow rate of user } x$$

2. **Rebalancing down**. The stable rate of a user x is rebalanced to the most recent value of B_s, if:

   $$B_s^x > B_s(1 + \Delta_{Bs})$$

   with Δ_{Bs} a rate delta established by governance which defines the window above B_s to rebalance interest rates. If a user pays too much interest beyond that range, the rate is balanced down.

4.5 The Rebalancing Process

The `LendingPool` contract exposes a function `rebalanceStableBorrowRate(address reserve, address_user)` which allows to rebalance the stable rate interest of a specific user. Anybody can call this function: however, there isn't any direct incentive for the caller to rebalance the rate of a specific user. For this reason, Aave will provide an agent that will periodically monitor all the stable rates positions and rebalance the ones that will be deemed necessary. The rebalance strategy will be decided offchain by the agent, this means that users that satisfy the rebalance conditions may not be rebalanced immediately. Since those conditions depend on the liquidity available and the state of market, there might be some transitory situations in which an immediate rebalance is not needed.

This does not add any element of centralization to the protocol. Even if the agent stops working, anybody can call the rebalance function of the `LendingPool` contract. Although there isn't any direct incentive in doing it ("why should I do it?") there is an indirect incentive for the ecosystem. In fact, even if the agent should cease to exist, depositors might still want to trigger a rebalance up of the lowest borrow rate positions, to increase the liquidity rate and/or force borrowers to close up their positions, increasing the available liquidity. In case of a rescale down, instead, borrowers have a direct incentive in performing a rebalance of their positions to lower the interest rate.

The following flowchart explains the sequence of actions of the function `rebalanceStableBorrowRate()`. The compounded balance that is accumulated until the instant at which the rebalance happens, is not affected by the rebalance.

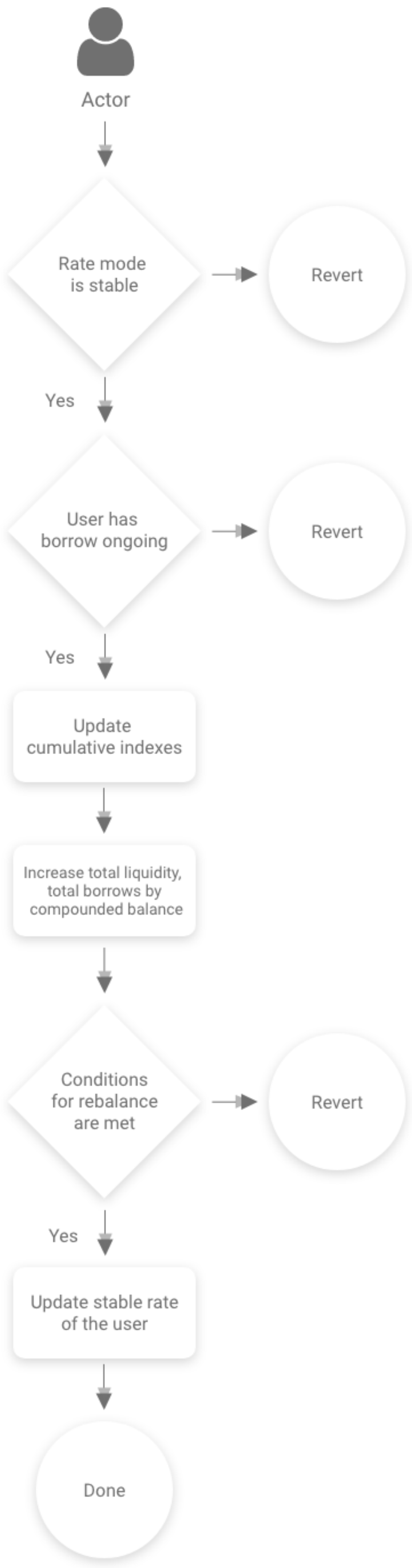

Actor
Rate mode is stable
Revert
Yes
User has borrow ongoing
Revert
Yes
Update cumulative indexes
Increase total liquidity, total borrows by compounded balance
Conditions for rebalance are met
Revert
Yes
Update stable rate of the user
Done

Figure 14: Rebalancing

5 Conclusion

The **Aave Protocol** relies on a lending pool model to offer high liquidity. Loans are backed by collateral and represented by aTokens, derivative tokens which accrue the interests. The parameters such as interest rate and Loan-To-Value are token specific.

Aave improves Decentralized Finance's current offering, bringing two key innovations to the lending ecosystem:

- **Stable Rates** to help borrowers' financial planning;
- **Flash Loans** to borrow without collateral during a single transaction.

Following the launch of the mainnet, Aave will uphold its commitment to decentralization through additional features. The Pool Factory will allow anyone to launch their own lending pool based on our smart-contracts. Governance will be on-chain with rights represented by:

- The LEND token at Protocol level for updates of the smart contract;
- aTokens at Pool level for pool specific parameters.

Curve DAO

Curve DAO consists of multiple smart contracts connected by Aragon. Apart from that, standard Aragon's 1 token = 1 vote method is replaced with the voting weight proportional to locktime, as will be described below.

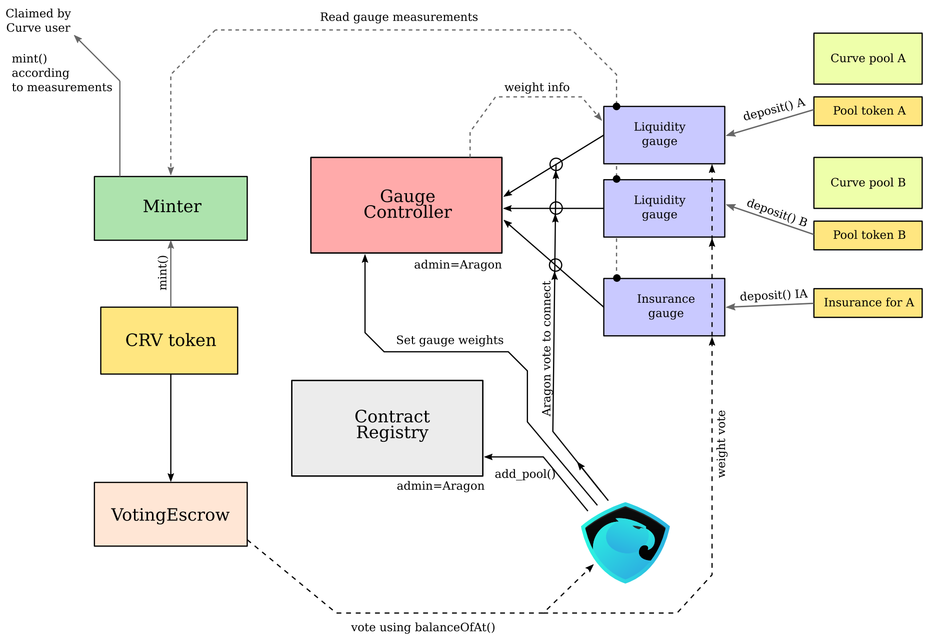


Figure 1: Curve DAO contracts managed by Aragon

Curve DAO has a token CRV which is used for both governance and value accrual.

Time-weighted voting. Vote-locked tokens in VotingEscrow

Instead of voting with token amount a, in Curve DAO tokens are lockable in a *VotingEscrow* for a selectable locktime t_l, where $t_l < t_{\max}$, and $t_{\max} = 4$ years. After locking, the time *left to unlock* is $t \leq t_l$. The voting weight is equal to:

$$w = a \frac{t}{t_{\max}}.$$

In other words, the vote is both amount- and time-weighted, where the time counted is how long the tokens cannot be moved in future.

The account which locks the tokens cannot be a smart contract (because can be tradable and/or tokenized), unless it is one of whitelisted smart contracts (for example, widely used multi-signature wallets).

VotingEscrow tries to resemble Aragon's Minime token. Most importantly, `balanceOf() / balanceOfAt()` and `totalSupply() / totalSupplyAt()` return the time-weighted voting weight w and the sum of all of those weights $W = \sum w_i$ respectively. Aragon can interface *VotingEscrow* as if it was a typical governance token.

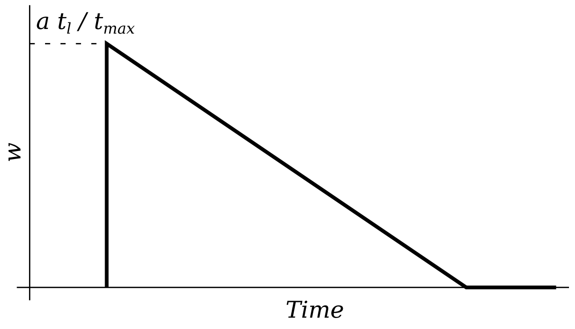


Figure 2: Voting weight of vote-locked tokens

Locks can be created with `create_lock()`, extended in time with `increase_unlock_time()` or token amount with `increase_amount()`, and `withdraw()` can remove tokens from the escrow when the lock is expired.

Implementation details

User voting power w_i is linearly decreasing since the moment of lock. So does the total voting power W. In order to avoid periodic check-ins, every time the user deposits, or withdraws, or changes the locktime, we *record user's slope and bias* for the linear function $w_i(t)$ in `user_point_history`. We also change slope and bias for the total voting power $W(t)$ and record in `point_history`. In addition, when user's lock is scheduled to end, we *schedule* change of slopes of $W(t)$ in the future in `slope_changes`. Every change involves increasing the `epoch` by 1.

This way we don't have to iterate over all users to figure out, how much should $W(t)$ change by, neither we require users to check in periodically. However, we limit the end of user locks to times rounded off by whole weeks.

Slopes and biases change both when a user deposits and locks governance tokens, and when the locktime expires. All the possible expiration times are rounded to whole weeks to make number of reads from blockchain proportional to number of missed weeks at most, not number of users (which can be potentially large).

Inflation schedule. ERC20CRV

Token *ERC20CRV* is an ERC20 token which allows a piecewise linear inflation schedule. The inflation is dropping by $2^{1/4}$ every year. Only *Minter* contract can directly mint *ERC20CRV*, but only within the limits defined by inflation.

Each time the inflation changes, a new mining epoch starts.

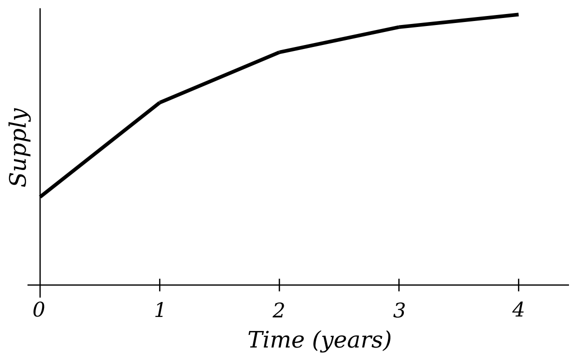


Figure 3: CRV token inflation schedule

Initial supply of CRV is 1.273 billion tokens, which is 42% of the eventual ($t \to \infty$) supply of $\approx$ 3.03 billion tokens. All of those initial tokens tokens are gradually vested (with every block). The initial inflation rate which supports the above inflation schedule is $r = 22.0\%$ (279.6 millions per year). All of the inflation is distributed to users of Curve, according to measurements taken by *gauges*. During the first year, the approximate inflow into circulating supply is 2 millions CRV per day, starting from 0.

System of Gauges. LiquidityGauge and GaugeController

In Curve, inflation is going towards users who use it. The usage is measured with Gauges. Currently there is just *LiquidityGauge* which measures, how much liquidity does the user provide. The same type of gauge can be used to measure "liquidity" provided for insurance.

For *LiquidityGauge* to measure user liquidity over time, the user deposits his LP tokens into the gauge using `deposit()` and can withdraw using `withdraw()`.

Coin rates which the gauge is getting depends on current inflation rate, and gauge *type weights* (which get voted on in Aragon). Each user gets inflation proportional to his LP tokens locked. Additionally, the rewards could be *boosted* by up to factor of 2.5 if user vote-locks tokens for Curve governance in *VotingEscrow*.

The user *does not* require to periodically check in. We describe how this is achieved in technical details.

GaugeController keeps a list of Gauges and their types, with weights of each gauge and type.

Gauges are per pool (each pool has an individual gauge).

LiquidityGauge implementation details

Suppose we have the inflation rate r changing with every epoch (1 year), gauge weight w_g and gauge type weight w_t. Then, all the gauge handles the stream of inflation with the rate $r' = w_g w_t r$ which it can update every time w_g, w_t, or mining epoch changes.

In order to calculate user's fair share of r', we essentially need to calculate the integral:

$$I_u = \int \frac{r'(t)\, b_u(t)}{S(t)}\, dt,$$

where $b_u(t)$ is the balance supplied by user (measured in LP tokens) and $S(t)$ is total liquidity supplied by users, depending on the time t; the value I_u gives the amount of tokens which user has to have minted to him. The user's balance b_u changes every time user u makes a deposit or withdrawal, and S changes every time *any* user makes a deposit or withdrawal (so S can change many times in between two events for the user u). In *LiquidityGauge* contract, the vaule of I_u is recorded in the `integrate_fraction` map, per-user.

In order to avoid all users to checkpoint periodically, we keep recording values of the following integral (named `integrate_inv_supply` in the contract):

$$I_{is}(t) = \int_0^t \frac{r'(t)}{S(t)} dt.$$

The value of I_{is} is recorded at any point any user deposits or withdraws, as well as every time the rate r' changes (either due to weight change or change of mining epoch).

When a user deposits or withdraws, the change in I_u can be calculated as the current (before user's action) value of I_{is} multiplied by the pre-action user's balance, and sumed up across user's balances:

$$I_u(t_k) = \sum_k b_u(t_k) \left[I_{is}(t_k) - I_{is}(t_{k-1}) \right].$$

The per-user integral is possible to repalce with this sum because $b_u(t)$ is unchanged for all times between t_{k-1} and t_k.

In order to incentivize users to participate in governance, and additionally create stickiness for liquidity, we implement the following mechanism. User's balance counted in the *LiquidityGauge* gets boosted by users locking CRV tokens in *VotingEscrow*, depending on their vote weight w_i:

$$b_u^* = \min\left(0.4\, b_u + 0.6\, S \frac{w_i}{W},\, b_u\right).$$

The value of w_i is taken at the time user performs any action (deposit, withdrawal, withdrawal of minted CRV tokens) and is applied until the next action this user performs.

If no users vote-lock any CRV (or simply don't have any), the inflation will simply be distributed proportionally to the liquidity b_u each one of them provided. However, if a user stakes much enough CRV, he is able to boost his stream of CRV by up to factor of 2.5 (reducing it slightly for all users who are not doing that).

Implementation details are such that a user gets the boost actual at the time of the last action or checkpoint. Since the voting power decreases with time, it is favorable for users to apply a boost and do no further actions until they vote-lock more tokens. However, once vote-lock expires, everyone can "kick" the user by creating a checkpoint for that user and, essentially, resetting the user to no boost if he/she has no voting power at that point already.

Finally, the gauge is supposed to not miss a full year of inflation (e.g. if there were no interactions with the guage for the full year). If that ever happens, the abandoned gauge gets less CRV.

Weight voting for gauges

Instead of simply voting for weight change in Aragon, users can allocate their vote-locked tokens towards one or other Gauge (pool). That pool will be getting a fraction of CRV tokens minted proportional to how much vote-locked tokens are allocated to it. Eeach user with tokens in VotingEscrow can change his/her preference at any time.

When a user applies a new weight vote, it gets applied only in the beginning of the next whole week (this is done for scalability reasons). The weight vote for the same gauge can be changed not more often than once in 10 days.

GaugeController implementation details

In order to implement weight voting, *GaugeController* has to include parameters handling linear character of voting power each user has.

Similarly to how it is done in *VotingEscrow*, *GaugeController* records points (bias+slope) per gauge in `vote_points`, *scheduled* changes in biases and slopes for those points in `vote_bias_changes` and `vote_slope_changes`, with those changes happening every round week, as well as current slopes for every user per-gauge in `vote_user_slopes`, along with the power the user has used and the time their vote-lock ends. The totals for slopes and biases for vote weight per gauge, and sums of those per type, get scheduled / recorded for the next week, as well as the points when voting power gets to 0 at lock expiration for some of users.

When user changes his preferences, the change of the gauge weight is scheduled for the next round week, not immediately. This is done in order to reduce the number of blockchain reads which need to be performed by each user: that will be proportional to the number of weeks since the last change instead of the number of interactions other users did.

GaugeController is one of the most central pieces to the system, so it must be controlled by the DAO. No centralized admin should control it, to not give anyone powers to change type weights unilaterally.

Fee burner

Every pool allows the admin to collect fees using `withdraw_admin_fees`. Aragon should be able to collect those fees to the admin account and use them to buy and burn CRV on a free market once that free market exists. That should be possible to be done by anyone without a vote.

Instead of burning, there could be different mechanisms working with the same interface. In any case, this will not be immediately applied.

Gauges to rewards trading volume and governance votes

Both votes and trades are discrete events, so they can use the same sort of gauge. The idea is that each event has a weight which exponentially decays over time.

It should be possible to call a gauge contract every time a user votes in Aragon.

Uniswap v3 Core

March 2021

Hayden Adams
hayden@uniswap.org

Noah Zinsmeister
noah@uniswap.org

Moody Salem
moody@uniswap.org

River Keefer
river@uniswap.org

Dan Robinson
dan@paradigm.xyz

ABSTRACT

Uniswap v3 is a noncustodial automated market maker implemented for the Ethereum Virtual Machine. In comparison to earlier versions of the protocol, Uniswap v3 provides increased capital efficiency and fine-tuned control to liquidity providers, improves the accuracy and convenience of the price oracle, and has a more flexible fee structure.

1 INTRODUCTION

Automated market makers (AMMs) are agents that pool liquidity and make it available to traders according to an algorithm [5]. Constant function market makers (CFMMs), a broad class of AMMs of which Uniswap is a member, have seen widespread use in the context of decentralized finance, where they are typically implemented as smart contracts that trade tokens on a permissionless blockchain [2].

CFMMs as they are implemented today are often capital inefficient. In the constant product market maker formula used by Uniswap v1 and v2, only a fraction of the assets in the pool are available at a given price. This is inefficient, particularly when assets are expected to trade close to a particular price at all times.

Prior attempts to address this capital efficiency issue, such as Curve [3] and YieldSpace [4], have involved building pools that use different functions to describe the relation between reserves. This requires all liquidity providers in a given pool to adhere to a single formula, and could result in liquidity fragmentation if liquidity providers want to provide liquidity within different price ranges.

In this paper, we present Uniswap v3, a novel AMM that gives liquidity providers more control over the price ranges in which their capital is used, with limited effect on liquidity fragmentation and gas inefficiency. This design does not depend on any shared assumption about the price behavior of the tokens. Uniswap v3 is based on the same constant product reserves curve as earlier versions [1], but offers several significant new features:

- *Concentrated Liquidity*: Liquidity providers (LPs) are given the ability to concentrate their liquidity by "bounding" it within an arbitrary price range. This improves the pool's capital efficiency and allows LPs to approximate their preferred reserves curve, while still being efficiently aggregated with the rest of the pool. We describe this feature in section 2 and its implementation in Section 6.
- *Flexible Fees*: The swap fee is no longer locked at 0.30%. Rather, the fee tier for each pool (of which there can be multiple per asset pair) is set on initialization (Section 3.1). The initially supported fee tiers are 0.05%, 0.30%, and 1%. `UNI` governance is able to add additional values to this set.
- *Protocol Fee Governance*: `UNI` governance has more flexibility in setting the fraction of swap fees collected by the protocol (Section 6.2.2).
- *Improved Price Oracle*: Uniswap v3 provides a way for users to query recent price accumulator values, thus avoiding the need to checkpoint the accumulator value at the exact beginning and end of the period for which a TWAP is being measured. (Section 5.1).

- *Liquidity Oracle*: The contracts expose a time-weighted average liquidity oracle (Section 5.3).

The Uniswap v2 core contracts are non-upgradeable by design, so Uniswap v3 is implemented as an entirely new set of contracts, available here. The Uniswap v3 core contracts are also non-upgradeable, with some parameters controlled by governance as described in Section 4.

2 CONCENTRATED LIQUIDITY

The defining idea of Uniswap v3 is that of *concentrated liquidity*: liquidity bounded within some price range.

In earlier versions, liquidity was distributed uniformly along the

$$x \cdot y = k \tag{2.1}$$

reserves curve, where x and y are the respective reserves of two assets X and Y, and k is a constant [1]. In other words, earlier versions were designed to provide liquidity across the entire price range $(0, \infty)$. This is simple to implement and allows liquidity to be efficiently aggregated, but means that much of the assets held in a pool are never touched.

Having considered this, it seems reasonable to allow LPs to concentrate their liquidity to smaller price ranges than $(0, \infty)$. We call liquidity concentrated to a finite range a *position*. A position only needs to maintain enough reserves to support trading within its range, and therefore can act like a constant product pool with larger reserves (we call these the *virtual reserves*) within that range.

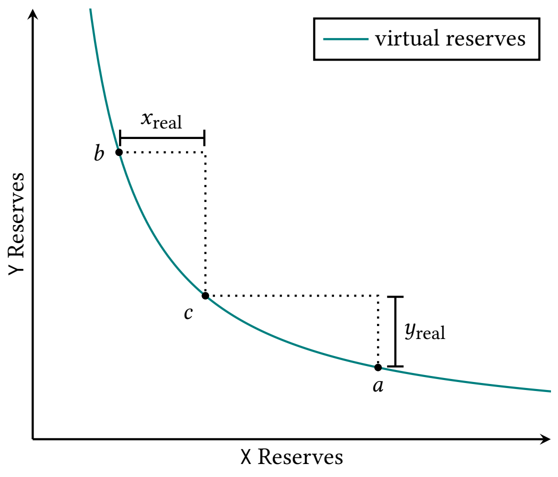


Figure 1: Simulation of Virtual Liquidity

Specifically, a position only needs to hold enough of asset X to cover price movement to its upper bound, because upwards price movement[1] corresponds to depletion of the X reserves. Similarly, it only needs to hold enough of asset Y to cover price movement to its lower bound. Fig. 1 depicts this relationship for a position on a range $[p_a, p_b]$ and a current price $p_c \in [p_a, p_b]$. x_{real} and y_{real} denote the position's real reserves.

When the price exits a position's range, the position's liquidity is no longer active, and no longer earns fees. At that point, its liquidity is composed entirely of a single asset, because the reserves of the other asset must have been entirely depleted. If the price ever reenters the range, the liquidity becomes active again.

The amount of liquidity provided can be measured by the value L, which is equal to $\sqrt{k}$. The real reserves of a position are described by the curve:

$$(x + \frac{L}{\sqrt{p_b}})(y + L\sqrt{p_a}) = L^2 \tag{2.2}$$

This curve is a translation of formula 2.1 such that the position is solvent exactly within its range (Fig. 2).

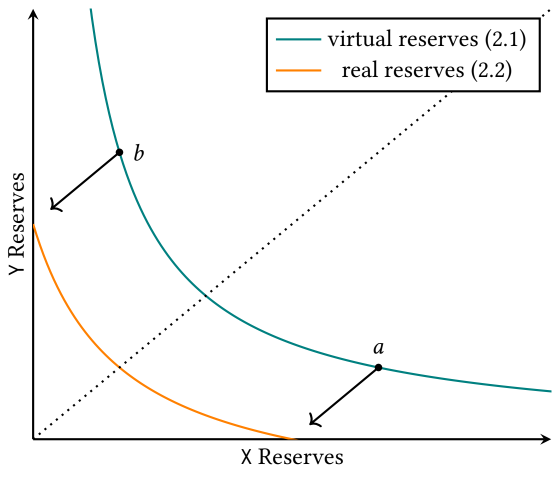


Figure 2: Real Reserves

Liquidity providers are free to create as many positions as they see fit, each on its own price range. In this way, LPs can approximate any desired distribution of liquidity on the price space (see Fig. 3 for a few examples). Moreover, this serves as a mechanism to let the market decide where liquidity should be allocated. Rational LPs can reduce their capital costs by concentrating their liquidity in a narrow band around the current price, and adding or removing tokens as the price moves to keep their liquidity active.

2.1 Range Orders

Positions on very small ranges act similarly to limit orders—if the range is crossed, the position flips from being composed entirely of one asset, to being composed entirely of the other asset (plus accrued fees). There are two differences between this *range order* and a traditional limit order:

- There is a limit to how narrow a position's range can be. While the price is within that range, the limit order might be partially executed.
- When the position has been crossed, it needs to be withdrawn. If it is not, and the price crosses back across that range, the position will be traded back, effectively reversing the trade.

[1]We take asset Y to be the unit of account, which corresponds to token1 in our implementation.

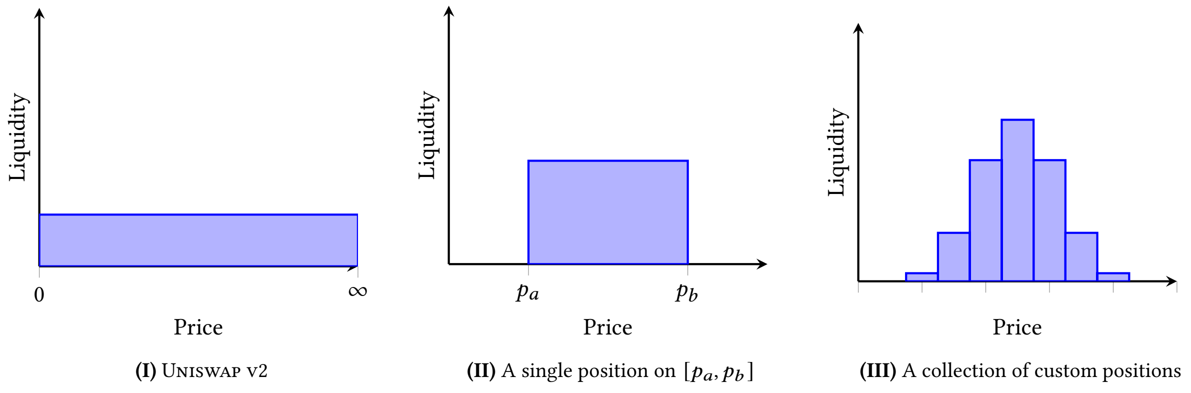


Figure 3: Example Liquidity Distributions

3 ARCHITECTURAL CHANGES

Uniswap v3 makes a number of architectural changes, some of which are necessitated by the inclusion of concentrated liquidity, and some of which are independent improvements.

3.1 Multiple Pools Per Pair

In Uniswap v1 and v2, every pair of tokens corresponds to a single liquidity pool, which applies a uniform fee of 0.30% to all swaps. While this default fee tier historically worked well enough for many tokens, it is likely too high for some pools (such as pools between two stablecoins), and too low for others (such as pools that include highly volatile or rarely traded tokens).

Uniswap v3 introduces multiple pools for each pair of tokens, each with a different swap fee. All pools are created by the same factory contract. The factory contract initially allows pools to be created at three fee tiers: 0.05%, 0.30%, and 1%. Additional fee tiers can be enabled by UNI governance.

3.2 Non-Fungible Liquidity

3.2.1 Non-Compounding Fees. Fees earned in earlier versions were continuously deposited in the pool as liquidity. This meant that liquidity in the pool would grow over time, even without explicit deposits, and that fee earnings compounded.

In Uniswap v3, due to the non-fungible nature of positions, this is no longer possible. Instead, fee earnings are stored separately and held as the tokens in which the fees are paid (see Section 6.2.2).

3.2.2 Removal of Native Liquidity Tokens. In Uniswap v1 and v2, the pool contract is also an ERC-20 token contract, whose tokens represent liquidity held in the pool. While this is convenient, it actually sits uneasily with the Uniswap v2 philosophy that anything that does not need to be in the core contracts should be in the periphery, and blessing one “canonical" ERC-20 implementation discourages the creation of improved ERC-20 token wrappers. Arguably, the ERC-20 token implementation should have been in the periphery, as a wrapper on a single liquidity position in the core contract.

The changes made in Uniswap v3 force this issue by making completely fungible liquidity tokens impossible. Due to the custom liquidity provision feature, fees are now collected and held by the pool as individual tokens, rather than automatically reinvested as liquidity in the pool.

As a result, in v3, the pool contract does not implement the ERC-20 standard. Anyone can create an ERC-20 token contract in the periphery that makes a liquidity position more fungible, but it will have to have additional logic to handle distribution of, or reinvestment of, collected fees. Alternatively, anyone could create a periphery contract that wraps an individual liquidity position (including collected fees) in an ERC-721 non-fungible token.

4 GOVERNANCE

The factory has an `owner`, which is initially controlled by UNI tokenholders.[2] The owner does not have the ability to halt the operation of any of the core contracts.

As in Uniswap v2, Uniswap v3 has a protocol fee that can be turned on by UNI governance. In Uniswap v3, UNI governance has more flexibility in choosing the fraction of swap fees that go to the protocol, and is able to choose any fraction $\frac{1}{N}$ where $4 \leq N \leq 10$, or 0. This parameter can be set on a per-pool basis.

UNI governance also has the ability to add additional fee tiers. When it adds a new fee tier, it can also define the `tickSpacing` (see Section 6.1) corresponding to that fee tier. Once a fee tier is added to the factory, it cannot be removed (and the `tickSpacing` cannot be changed). The initial fee tiers and tick spacings supported are 0.05% (with a tick spacing of 10, approximately 0.10% between initializable ticks), 0.30% (with a tick spacing of 60, approximately 0.60% between initializable ticks), and 1% (with a tick spacing of 200, approximately 2.02% between ticks.

Finally, UNI governance has the power to transfer ownership to another address.

5 ORACLE UPGRADES

Uniswap v3 includes three significant changes to the time-weighted average price (TWAP) oracle that was introduced by Uniswap v2.

Most significantly, Uniswap v3 removes the need for users of the oracle to track previous values of the accumulator externally. Uniswap v2 requires users to checkpoint the accumulator value at both the beginning and end of the time period for which they

[2]Specifically, the owner will be initialized to the Timelock contract from UNI governance, 0x1a9c8182c09f50c8318d769245bea52c32be35bc.

wanted to compute a TWAP. Uniswap v3 brings the accumulator checkpoints into core, allowing external contracts to compute on-chain TWAPs over recent periods without storing checkpoints of the accumulator value.

Another change is that instead of accumulating the sum of prices, allowing users to compute the arithmetic mean TWAP, Uniswap v3 tracks the sum of *log* prices, allowing users to compute the *geometric mean* TWAP.

Finally, Uniswap v3 adds a liquidity accumulator that is tracked alongside the price accumulator. This liquidity accumulator can be used by other contracts to inform a decision on which of the pools corresponding to a pair (see section 3.1) will have the most reliable TWAP.

5.1 Oracle Observations

As in Uniswap v2, Uniswap v3 tracks a running accumulator of the price at the beginning of each block, multiplied by the number of seconds since the last block.

A pool in Uniswap v2 stores only the most recent value of this price accumulator—that is, the value as of the last block in which a swap occurred. When computing average prices in Uniswap v2, it is the responsibility of the external caller to provide the previous value of the price accumulator. With many users, each will have to provide their own methodology for checkpointing previous values of the accumulator, or coordinate on a shared method to reduce costs. And there is no way to guarantee that every block in which the pool is touched will be reflected in the accumulator.

In Uniswap v3, the pool stores a list of previous values for the accumulator. It does this by automatically checkpointing the accumulator value every time the pool is touched for the first time in a block, cycling through an array where the oldest checkpoint is eventually overwritten by a new one, similar to a circular buffer. While this array initially only has room for a single checkpoint, anyone can initialize additional storage slots to lengthen the array, extending to as many as 65,536 checkpoints.[3] This imposes the one-time gas cost of initializing additional storage slots for this array on whoever wants this pair to checkpoint more slots.

The pool exposes the array of past observations to users, as well as a convenience function for finding the (interpolated) accumulator value at any historical timestamp within the checkpointed period.

5.2 Geometric Mean Price Oracle

Uniswap v2 maintains two price accumulators—one for the price of `token0` in terms of token1, and one for the price of `token1` in terms of token0. Users can compute the time-weighted arithmetic mean of the prices over any period, by subtracting the accumulator value at the beginning of the period from the accumulator at the end of the period, then dividing the difference by the number of seconds in the period. Note that accumulators for `token0` and `token1` are tracked separately, since the time-weighted arithmetic mean price of `token0` is not equivalent to the reciprocal of the time-weighted arithmetic mean price of `token1`.

Using the time-weighted *geometric* mean price, as Uniswap v3 does, avoids the need to track separate accumulators for these ratios. The geometric mean of a set of ratios is the reciprocal of the geometric mean of their reciprocals. It is also easy to implement in Uniswap v3 because of its implementation of custom liquidity provision, as described in section 6. In addition, the accumulator can be stored in a smaller number of bits, since it tracks $\log P$ rather than P, and $\log P$ can represent a wide range of prices with consistent precision.[4] Finally, there is a theoretical argument that the time-weighted geometric mean price should be a truer representation of the average price.[5]

Instead of tracking the cumulative sum of the price P, Uniswap v3 accumulates the cumulative sum of the current tick index ($log_{1.0001}P$, the logarithm of price for base 1.0001, which is precise up to 1 basis point). The accumulator at any given time is equal to the sum of $log_{1.0001}(P)$ for every second in the history of the contract:

$$a_t = \sum_{i=1}^{t} \log_{1.0001}(P_i) \tag{5.1}$$

We want to estimate the geometric mean time-weighted average price (p_{t_1,t_2}) over any period t_1 to t_2.

$$P_{t_1,t_2} = \left(\prod_{i=t_1}^{t_2} P_i\right)^{\frac{1}{t_2-t_1}} \tag{5.2}$$

To compute this, you can look at the accumulator's value at t_1 and at t_2, subtract the first value from the second, divide by the number of seconds elapsed, and compute 1.0001^x to compute the time weighted geometric mean price.

$$\log_{1.0001}\left(P_{t_1,t_2}\right) = \frac{\sum_{i=t_1}^{t_2} \log_{1.0001}(P_i)}{t_2 - t_1} \tag{5.3}$$

$$\log_{1.0001}\left(P_{t_1,t_2}\right) = \frac{a_{t_2} - a_{t_1}}{t_2 - t_1} \tag{5.4}$$

$$P_{t_1,t_2} = 1.0001^{\frac{a_{t_2}-a_{t_1}}{t_2-t_1}} \tag{5.5}$$

5.3 Liquidity Oracle

In addition to the time weighted average price, Uniswap v3 also tracks an accumulator of the current value of L (the virtual liquidity currently in range) at the beginning of each block. This can be used by on-chain contracts to make their oracles stronger (such as by evaluating which fee-tier pool to use the oracle from). This liquidity accumulator's values are checkpointed along with the price accumulator.

[3]The maximum of 65,536 checkpoints allows fetching checkpoints for at least 9 days after they are written, assuming 13 seconds pass between each block and a checkpoint is written every block.

[4]In order to support tolerable precision across all possible prices, Uniswap v2 represents each price as a 224-bit fixed-point number. Uniswap v3 only needs to represent $log_{1.0001}P$ as a signed 24-bit number, and still can detect price movements of one tick, or 1 basis point.

[5]While arithmetic mean TWAPs are much more widely used, they should theoretically be less accurate in measuring a geometric Brownian motion process (which is how price movements are usually modeled). The arithmetic mean of a geometric Brownian motion process will tend to overweight higher prices (where small percentage movements correspond to large absolute movements) relative to lower ones.

6 IMPLEMENTING CONCENTRATED LIQUIDITY

The rest of this paper describes how concentrated liquidity provision works, and gives a high-level description of how it is implemented in the contracts.

6.1 Ticks and Ranges

To implement custom liquidity provision, the space of possible prices is demarcated by discrete *ticks*. Liquidity providers can provide liquidity in a range between any two ticks (which need not be adjacent).

Each range can be specified as a pair of signed integer *tick indices*: a lower tick (i_l) and an upper tick (i_u). Ticks represent prices at which the virtual liquidity of the contract can change. We will assume that prices are always expressed as the price of one of the tokens—called `token0`—in terms of the other token—`token1`. The assignment of the two tokens to `token0` and `token1` is arbitrary and does not affect the logic of the contract (other than through possible rounding errors).

Conceptually, there is a tick at every price p that is an integer power of 1.0001. Identifying ticks by an integer index i, the price at each is given by:

$$p(i) = 1.0001^i \tag{6.1}$$

This has the desirable property of each tick being a .01% (1 basis point) price movement away from each of its neighboring ticks.

For technical reasons explained in 6.2.1, however, pools actually track ticks at every *square root price* that is an integer power of $\sqrt{1.0001}$. Consider the above equation, transformed into square root price space:

$$\sqrt{p}(i) = \sqrt{1.0001}^i = 1.0001^{\frac{i}{2}} \tag{6.2}$$

As an example, $\sqrt{p}(0)$—the square root price at tick 0—is 1, $\sqrt{p}(1)$ is $\sqrt{1.0001} \approx 1.00005$, and $\sqrt{p}(-1)$ is $\frac{1}{\sqrt{1.0001}} \approx 0.99995$.

When liquidity is added to a range, if one or both of the ticks is not already used as a bound in an existing position, that tick is *initialized*.

Not every tick can be initialized. The pool is instantiated with a parameter, `tickSpacing` (t_s); only ticks with indexes that are divisible by `tickSpacing` can be initialized. For example, if `tickSpacing` is 2, then only even ticks (...-4, -2, 0, 2, 4...) can be initialized. Small choices for `tickSpacing` allow tighter and more precise ranges, but may cause swaps to be more gas-intensive (since each initialized tick that a swap crosses imposes a gas cost on the swapper).

Whenever the price crosses an initialized tick, virtual liquidity is kicked in or out. The gas cost of an initialized tick crossing is constant, and is not dependent on the number of positions being kicked in or out at that tick.

Ensuring that the right amount of liquidity is kicked in and out of the pool when ticks are crossed, and ensuring that each position earns its proportional share of the fees that were accrued while it was within range, requires some accounting within the pool. The pool contract uses storage variables to track state at a *global* (per-pool) level, at a *per-tick* level, and at a *per-position* level.

6.2 Global State

The global state of the contract includes seven storage variables relevant to swaps and liquidity provision. (It has other storage variables that are used for the oracle, as described in section 5.)

Type	Variable Name	Notation
`uint128`	`liquidity`	L
`uint160`	`sqrtPriceX96`	$\sqrt{P}$
`int24`	`tick`	i_c
`uint256`	`feeGrowthGlobal0X128`	$f_{g,0}$
`uint256`	`feeGrowthGlobal1X128`	$f_{g,1}$
`uint128`	`protocolFees.token0`	$f_{p,0}$
`uint128`	`protocolFees.token1`	$f_{p,1}$

Table 1: Global State

6.2.1 Price and Liquidity. In Uniswap v2, each pool contract tracks the pool's current reserves, x and y. In Uniswap v3, the contract could be thought of as having *virtual reserves*—values for x and y that allow you to describe the contract's behavior (between two adjacent ticks) as if it followed the constant product formula.

Instead of tracking those virtual reserves, however, the pool contract tracks two different values: `liquidity` (L) and `sqrtPrice` ($\sqrt{P}$). These could be computed from the virtual reserves with the following formulas:

$$L = \sqrt{xy} \tag{6.3}$$

$$\sqrt{P} = \sqrt{\frac{y}{x}} \tag{6.4}$$

Conversely, these values could be used to compute the virtual reserves:

$$x = \frac{L}{\sqrt{P}} \tag{6.5}$$

$$y = L \cdot \sqrt{P} \tag{6.6}$$

Using L and $\sqrt{P}$ is convenient because only one of them changes at a time. Price (and thus $\sqrt{P}$) changes when swapping within a tick; liquidity changes when crossing a tick, or when minting or burning liquidity. This avoids some rounding errors that could be encountered if tracking virtual reserves.

You may notice that the formula for liquidity (based on virtual reserves) is similar to the formula used to initialize the quantity of liquidity tokens (based on actual reserves) in Uniswap v2. before any fees have been earned. In some ways, liquidity can be thought of as virtual liquidity tokens.

Alternatively, liquidity can be thought of as the amount that `token1` reserves (either actual or virtual) changes for a given change in $\sqrt{P}$:

$$L = \frac{\Delta Y}{\Delta \sqrt{P}} \tag{6.7}$$

We track $\sqrt{P}$ instead of P to take advantage of this relationship, and to avoid having to take any square roots when computing swaps, as described in section 6.2.3.

The global state also tracks the current tick index as `tick` (i_c), a signed integer representing the current tick (more specifically, the nearest tick below the current price). This is an optimization (and a way of avoiding precision issues with logarithms), since at any time, you should be able to compute the current tick based on the current `sqrtPrice`. Specifically, at any given time, the following equation should be true:

$$i_c = \left\lfloor \log_{\sqrt{1.0001}} \sqrt{P} \right\rfloor \tag{6.8}$$

6.2.2 Fees. Each pool is initialized with an immutable value, `fee` (γ), representing the fee paid by swappers in units of hundredths of a basis point (0.0001%).

It also tracks the current protocol fee, ϕ (which is initialized to zero, but can changed by UNI governance).[6] This number gives you the fraction of the fees paid by swappers that currently goes to the protocol rather than to liquidity providers. ϕ only has a limited set of permitted values: 0, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9, or 1/10.

The global state also tracks two numbers: `feeGrowthGlobal0` ($f_{g,0}$) and `feeGrowthGlobal1` ($f_{g,1}$). These represent the total amount of fees that have been earned per unit of virtual liquidity (L), over the entire history of the contract. You can think of them as the total amount of fees that would have been earned by 1 unit of unbounded liquidity that was deposited when the contract was first initialized. They are stored as fixed-point unsigned 128x128 numbers. Note that in Uniswap v3, fees are collected in the tokens themselves rather than in liquidity, for reasons explained in section 3.2.1.

Finally, the global state tracks the total accumulated uncollected protocol fee in each token, `protocolFees0` ($f_{p,0}$) and `protocolFees1` ($f_{p,1}$). This is an unsigned `uint128`. The accumulated protocol fees can be collected by UNI governance, by calling the `collectProtocol` function.

6.2.3 Swapping Within a Single Tick. For small enough swaps, that do not move the price past a tick, the contracts act like an $x \cdot y = k$ pool.

Suppose γ is the fee, i.e., 0.003, and y_{in} as the amount of `token1` sent in.

First, `feeGrowthGlobal1` and `protocolFees1` are incremented:

$$\Delta f_{g,1} = y_{in} \cdot \gamma \cdot (1 - \phi) \tag{6.9}$$

$$\Delta f_{p,1} = y_{in} \cdot \gamma \cdot \phi \tag{6.10}$$

Δy is the increase in y (after the fee is taken out).

$$\Delta y = y_{in} \cdot (1 - \gamma) \tag{6.11}$$

If you used the computed virtual reserves (x and y) for the `token0` and `token1` balances, then this formula could be used to find the amount of `token0` sent out:

$$x_{end} = \frac{x \cdot y}{y + \Delta y} \tag{6.12}$$

But remember that in v3, the contract actually tracks liquidity (L) and square root of price ($\sqrt{P}$) instead of x and y. We could compute x and y from those values, and then use those to calculate the execution price of the trade. But it turns out that there are simple formulas that describe the relationship between $\Delta\sqrt{P}$ and Δy, for a given L (which can be derived from formula 6.7):

$$\Delta\sqrt{P} = \frac{\Delta y}{L} \tag{6.13}$$

$$\Delta y = \Delta\sqrt{P} \cdot L \tag{6.14}$$

There are also simple formulas that describe the relationship between $\Delta\frac{1}{\sqrt{P}}$ and Δx:

$$\Delta\frac{1}{\sqrt{P}} = \frac{\Delta x}{L} \tag{6.15}$$

$$\Delta x = \Delta\frac{1}{\sqrt{P}} \cdot L \tag{6.16}$$

When swapping one token for the other, the pool contract can first compute the new $\sqrt{P}$ using formula 6.13 or 6.15, and then can compute the amount of `token0` or `token1` to send out using formula 6.14 or 6.16.

These formulas will work for any swap that does not push $\sqrt{P}$ past the price of the next initialized tick. If the computed $\Delta\sqrt{P}$ would cause $\sqrt{P}$ to move past that next initialized tick, the contract must only cross up to that tick—using up only part of the swap—and then cross the tick, as described in section 6.3.1, before continuing with the rest of the swap.

6.2.4 Initialized Tick Bitmap. If a tick is not used as the endpoint of a range with any liquidity in it—that is, if the tick is uninitialized—then that tick can be skipped during swaps.

As an optimization to make finding the next initialized tick more efficient, the pool tracks a bitmap `tickBitmap` of initialized ticks. The position in the bitmap that corresponds to the tick index is set to 1 if the tick is initialized, and 0 if it is not initialized.

When a tick is used as an endpoint for a new position, and that tick is not currently used by any other liquidity, the tick is initialized, and the corresponding bit in the bitmap is set to 1. An initialized tick can become uninitialized again if all of the liquidity for which it is an endpoint is removed, in which case that tick's position on the bitmap is zeroed out.

6.3 Tick-Indexed State

The contract needs to store information about each tick in order to track the amount of net liquidity that should be added or removed when the tick is crossed, as well as to track the fees earned above and below that tick.

The contract stores a mapping from tick indexes (`int24`) to the following four values:

Type	Variable Name	Notation
int128	liquidityNet	ΔL
uint128	liquidityGross	L_g
uint256	feeGrowthOutside0X128	$f_{o,0}$
uint256	feeGrowthOutside1X128	$f_{o,1}$

Table 2: Tick-Indexed State

[6] Technically, the storage variable called "protocolFee" is the denominator of this fraction (or is zero, if ϕ is zero).

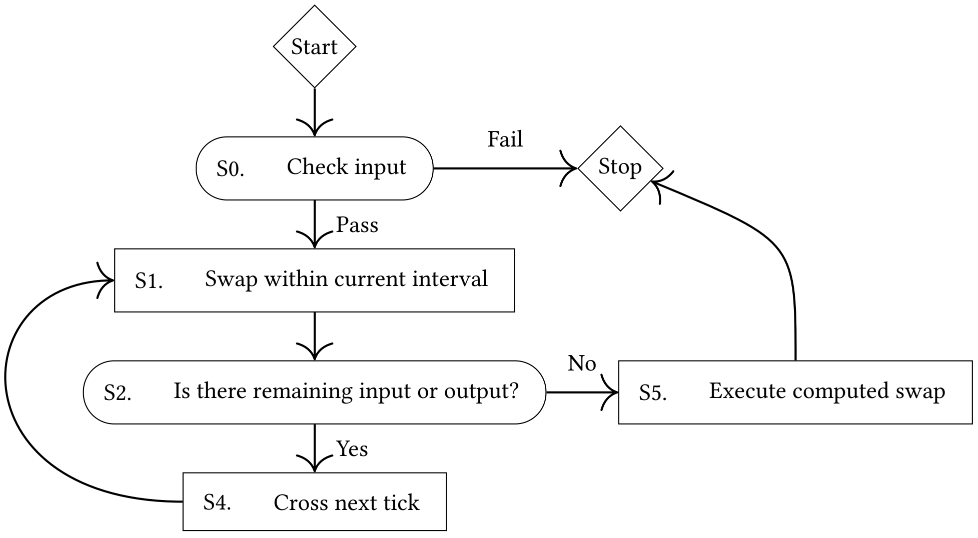

Start
S0. Check input
Fail
Stop
Pass
S1. Swap within current interval
S2. Is there remaining input or output?
No
S5. Execute computed swap
Yes
S4. Cross next tick

Figure 4: Swap Control Flow

Each tick tracks ΔL, the total amount of liquidity that should be kicked in or out when the tick is crossed. The tick only needs to track one signed integer: the amount of liquidity added (or, if negative, removed) when the tick is crossed going left to right. This value does not need to be updated when the tick is crossed (but only when a position with a bound at that tick is updated).

We want to be able to uninitialize a tick when there is no longer any liquidity referencing that tick. Since ΔL is a net value, it's necessary to track a gross tally of liquidity referencing the tick, `liquidityGross`. This value ensures that even if net liquidity at a tick is 0, we can still know if a tick is referenced by at least one underlying position or not, which tells us whether to update the tick bitmap.

`feeGrowthOutside{0,1}` are used to track how many fees were accumulated within a given range. Since the formulas are the same for the fees collected in `token0` and token1, we will omit that subscript for the rest of this section.

You can compute the fees earned per unit of liquidity in token 0 above (f_a) and below (f_b) a tick i with a formula that depends on whether the price is currently within or outside that range—that is, whether the current tick index i_c is greater than or equal to i:

$$f_a(i) = \begin{cases} f_g - f_o(i) & i_c \geq i \\ f_o(i) & i_c < i \end{cases} \tag{6.17}$$

$$f_b(i) = \begin{cases} f_o(i) & i_c \geq i \\ f_g - f_o(i) & i_c < i \end{cases} \tag{6.18}$$

We can use these functions to compute the total amount of cumulative fees per share f_{i_l,i_u} in the range between two ticks—a lower tick i_l and an upper tick i_u:

$$f_{i_l,i_u}(0) = f_g - f_b(i_l) - f_a(i_u) \tag{6.19}$$

$f_{o,1}$ needs to be updated each time the tick is crossed. Specifically, as a tick i is crossed in either direction, its f_o (for each token) should be updated as follows:

$$f_o(i) := f_g - f_o(i) \tag{6.20}$$

f_o is only needed for ticks that are used as either the lower or upper bound for at least one position. As a result, for efficiency, f_o is not initialized (and thus does not need to be updated when crossed) until a position is created that has that tick as one of its bounds. When f_o is initialized for a tick i, the value—by convention—is chosen as if all of the fees earned to date had occurred below that tick:

$$f_o := \begin{cases} f_g & i_c \geq i \\ 0 & i_c < i \end{cases} \tag{6.21}$$

Note that since f_o values for different ticks could be initialized at different times, comparisons of the f_o values for different ticks are not meaningful, and there is no guarantee that values for f_o will be consistent. This does not cause a problem for per-position accounting, since, as described below, all the position needs to know is the growth in g within a given range since that position was last touched.

Finally, the contract also stores `secondsOutside` (t_o) for each tick. This can be thought of as the amount of time spent on the other side of this tick (relative to the current price), and can be used to compute the number of seconds that have been spend above or below a tick for a particular range. This value is not used within the contract, but is tracked for the convenience of external contracts that want to know how many seconds a given position has been active.

As with f_a and f_b, the time spent above (t_a) and below (t_b) a given tick is computed differently based on whether the current price is within that range, and the time spent within a range (t_r) can be computed using the values of t_a and t_b

$$t_a(i) = \begin{cases} t - t_o(i) & i_c \geq i \\ t_o(i) & i_c < i \end{cases} \tag{6.22}$$

$$t_b(i) = \begin{cases} t_o(i) & i_c \geq i \\ t - t_o(i) & i_c < i \end{cases} \tag{6.23}$$

$$t_r(i_l, i_u) = t - t_b(i_l) - t_a(i_u) \tag{6.24}$$

The number of seconds spent within a range between two times t_1 and t_2 can be computed by recording the value of $t_r(i_l, i_u)$ at t_1 and at t_2, and subtracting the former from the latter.

Like f_o, t_o does not need to be tracked for ticks that are not on the edge of any position. Therefore, it is not initialized until a position is created that is bounded by that tick. By convention, it is initialized as if every second since the Unix timestamp 0 had been spent below that tick:

$$t_o(i) := \begin{cases} t & i_c \geq i \\ 0 & i_c < i \end{cases} \tag{6.25}$$

As with f_o values, t_o values are not meaningfully comparable across different ticks. t_o is only meaningful in computing the number of seconds that liquidity was within some particular range between some defined start time (which must be after t_o was initialized for both ticks) and some end time.

6.3.1 Crossing a Tick. As described in section 6.2.3, Uniswap v3 acts like it obeys the constant product formula when swapping between initialized ticks. When a swap crosses an initialized tick, however, the contract needs to add or remove liquidity, to ensure that no liquidity provider is insolvent. This means the ΔL is fetched from the tick, and applied to the global L.

The contract also needs to update the tick's own state, in order to track the fees earned (and seconds spent) within ranges bounded by this tick. The `feeGrowthOutside{0,1}` and `secondsOutside` values are updated to both reflect current values, as well as the proper orientation relative to the current tick:

$$f_o := f_g - f_o \tag{6.26}$$

$$t_o := t - t_o \tag{6.27}$$

Once a tick is crossed, the swap can continue as described in section 6.2.3 until it reaches the next initialized tick.

6.4 Position-Indexed State

The contract has a mapping from user (an `address`), lower bound (a tick index, `int24`), and upper bound (a tick index, `int24`) to a specific `Position` struct. Each `Position` tracks three values:

Type	Variable Name	Notation
`uint128`	`liquidity`	l
`uint256`	`feeGrowthInside0LastX128`	$f_{r,0}(t_0)$
`uint256`	`feeGrowthInside1LastX128`	$f_{r,1}(t_0)$

Table 3: Position-Indexed State

`liquidity` (l) means the amount of virtual liquidity that the position represented the last time this position was touched. Specifically, `liquidity` could be thought of as $\sqrt{x \cdot y}$, where x and y are the respective amounts of virtual `token0` and virtual `token1` that this liquidity contributes to the pool at any time that it is within range. Unlike pool shares in Uniswap v2 (where the value of each share grows over time), the units for liquidity do not change as fees are accumulated; it is always measured as $\sqrt{x \cdot y}$, where x and y are quantities of `token0` and token1, respectively.

This `liquidity` number does not reflect the fees that have been accumulated since the contract was last touched, which we will call *uncollected fees*. Computing these uncollected fees requires additional stored values on the position, `feeGrowthInside0Last` ($f_{r,0}(t_0)$) and `feeGrowthInside1Last` ($f_{r,1}(t_0)$), as described below.

6.4.1 setPosition. The `setPosition` function allows a liquidity provider to update their position.

Two of the arguments to `setPosition` —`lowerTick` and `upperTick`— when combined with the `msg.sender`, together specify a position.

The function takes one additional parameter, `liquidityDelta`, to specify how much virtual liquidity the user wants to add or (if negative) remove.

First, the function computes the uncollected fees (f_u) that the position is entitled to, in each token.[7] The amount collected in fees is credited to the user and netted against the amount that they would send in or out for their virtual liquidity deposit.

To compute uncollected fees of a token, you need to know how much f_r for the position's range (calculated from the range's i_l and i_r as described in section 6.3) has grown since the last time fees were collected for that position. The growth in fees in a given range per unit of liquidity over between times t_0 and t_1 is simply $f_r(t_1) - f_r(t_0)$ (where $f_r(t_0)$ is stored in the position as `feeGrowthInside{0,1}Last`, and $f_r(t_1)$ can be computed from the current state of the ticks). Multiplying this by the position's `liquidity` gives us the total uncollected fees in token 0 for this position:

$$f_u = l \cdot (f_r(t_1) - f_r(t_0)) \tag{6.28}$$

Then, the contract updates the position's `liquidity` by adding `liquidityDelta`. It also adds `liquidityDelta` to the `liquidityNet` value for the tick at the bottom end of the range, and subtracts it from the `liquidityNet` at the upper tick (to reflect that this new liquidity would be added when the price crosses the lower tick going up, and subtracted when the price crosses the upper tick going up). If the pool's current price is within the range of this position, the contract also adds `liquidityDelta` to the contract's global `liquidity` value.

Finally, the pool transfers tokens from (or, if `liquidityDelta` is negative, to) the user, corresponding to the amount of liquidity burned or minted.

The amount of `token0` (ΔX) or `token1` (ΔY) that needs to be deposited can be thought of as the amount that would be sold from the position if the price were to move from the current price (P) to the upper tick or lower tick (for `token0` or `token1`, respectively). These formulas can be derived from formulas 6.14 and 6.16, and depend on whether the current price is below, within, or above the range of the position:

[7] Since the formulas for computing uncollected fees in each token are the same, we will omit that subscript for the rest of this section.

$$\Delta Y = \begin{cases} 0 & i_c < i_l \\ \Delta L \cdot (\sqrt{P} - \sqrt{p(i_l)}) & i_l \leq i_c < i_u \\ \Delta L \cdot (\sqrt{p(i_u)} - \sqrt{p(i_l)}) & i_c \geq i_u \end{cases} \tag{6.29}$$

$$\Delta X = \begin{cases} \Delta L \cdot (\frac{1}{\sqrt{p(i_l)}} - \frac{1}{\sqrt{p(i_u)}}) & i_c < i_l \\ \Delta L \cdot (\frac{1}{\sqrt{P}} - \frac{1}{\sqrt{p(i_u)}}) & i_l \leq i_c < i_u \\ 0 & i_c \geq i_u \end{cases} \tag{6.30}$$

REFERENCES

[1] Hayden Adams, Noah Zinsmeister, and Dan Robinson. 2020. *Uniswap v2 Core.* Retrieved Feb 24, 2021 from https://uniswap.org/whitepaper.pdf

[2] Guillermo Angeris and Tarun Chitra. 2020. Improved Price Oracles: Constant Function Market Makers. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFT '20).* Association for Computing Machinery, New York, NY, United States, 80–91. https://doi.org/10.1145/3419614.3423251

[3] Michael Egorov. 2019. *StableSwap - Efficient Mechanism for Stablecoin Liquidity.* Retrieved Feb 24, 2021 from https://www.curve.fi/stableswap-paper.pdf

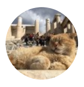
[4] Allan Niemerg, Dan Robinson, and Lev Livnev. 2020. *YieldSpace: An Automated Liquidity Provider for Fixed Yield Tokens.* Retrieved Feb 24, 2021 from https://yield.is/YieldSpace.pdf

[5] Abraham Othman. 2012. *Automated Market Making: Theory and Practice.* Ph.D. Dissertation. Carnegie Mellon University.

DISCLAIMER

DeFi Smart Accounts

Scaling DeFi for users, wallets, developers and protocols



MING
Published on Mar 1

SHARE:

In 2019, Instadapp achieved a position of 3rd in ETH locked value with a deep focus on improving the usability of DeFi. We did so over many key innovations, including DeFi smart wallets, simplifying cross-protocol complex transactions, and portability between protocols and stablecoins.

Moving into 2020 and beyond, we believe that building better user experiences in DeFi and making it easier for developers to build secure dApps remains the primary criterion for bringing this space closer to the mainstream. However, there remain several fundamental challenges on both fronts.

Fundamental UX and Security Challenges

Ethereum accounts and wallets were designed primarily for token assets, while **DeFi assets have a much higher level of complexity in terms of management, admin, analysis**. This requires a completely different approach to UX, portfolio management, and admin delegation. This problem will only become more severe as the number of underlying protocols grows exponentially.

When asked to manage all their assets themselves, or under a single account, users are likely to lose their assets, trust insecure dApps, or provide allowances to many smart contracts. Developers, particularly those with more frontend and server experience, are critical for designing the use cases for mainstream users. Still, they might not have the required expertise to ensure security on on-chain applications.

Instadapp DeFi Accounts

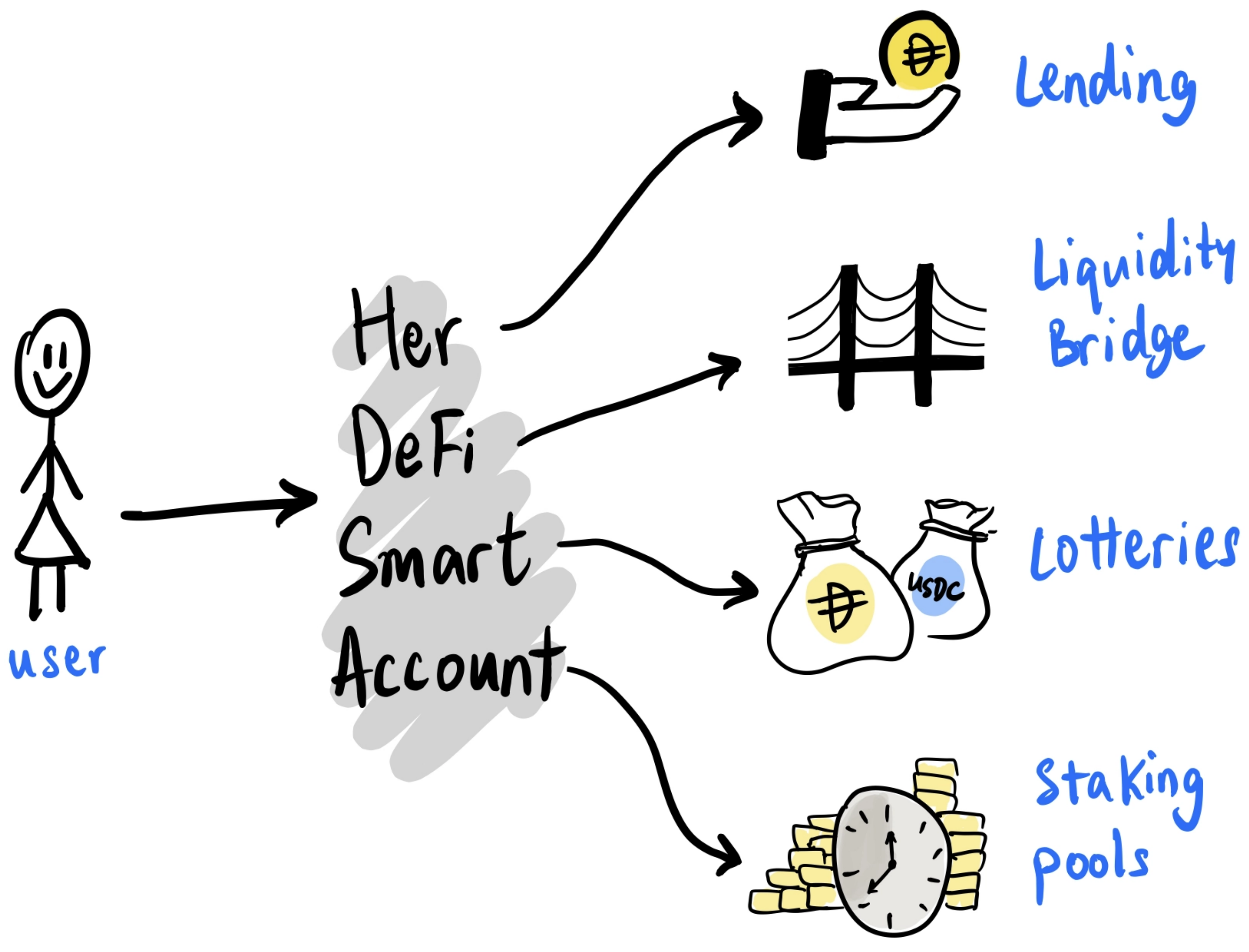
To tackle these issues, we are announcing DeFi Accounts, a platform providing users and developers with a single point of integration to access all the DeFi protocols. We aim to make DeFi **easier, scalable, and more secure for users, developers, and protocols alike**.

Users	Trustlessly manage, delegate, and optimize funds across any number of protocols.
Developers	Innovate interfaces and business models on top of DeFi accounts. Securely compose complex transactions without smart contracts.
Protocols	Allow your system to be easily accessed by new users and developers.

We would also like to invite everyone to join our user group or our brand new developer group. The latter is brand new so that you will get all the attention. 😍

For DeFi Users

1) Ability To Use One Platform For Their DeFi Needs

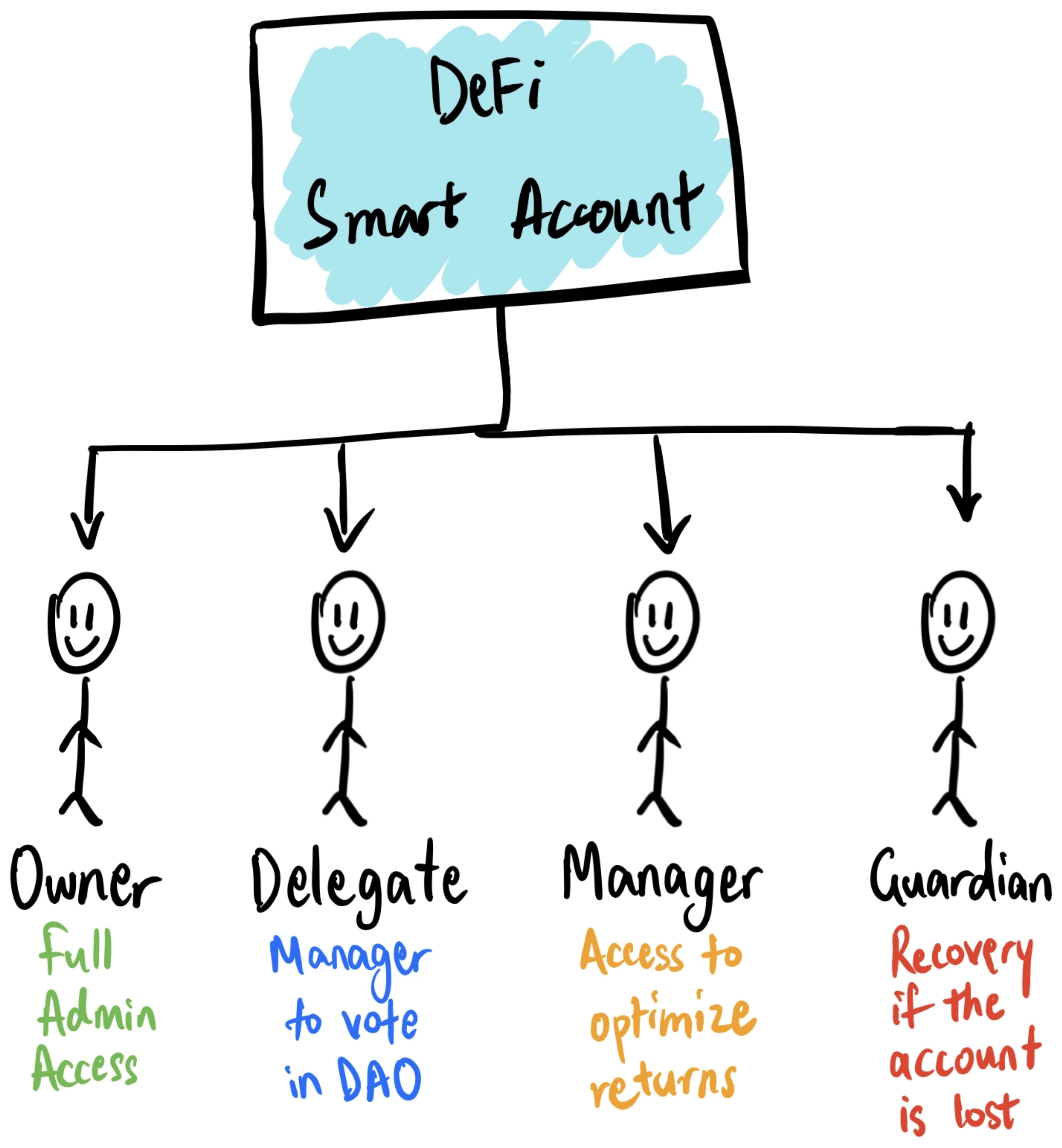


Today, smart wallets have the potential to improve the usability of DeFi. However, they are too limited in scope and usefulness to replace main Ethereum accounts since customized proxies have to be built for every single new use case.

In contrast, once new protocols are connected to DeFi smart accounts with connectors, any developer can then easily extend their product to include cross-protocol transactions for those protocols by writing simple Web2 code without new smart contracts. This also leads to a much higher level of usability and security for users.

Building on DeFi smart accounts, developers (including Instadapp and wallets) can build powerful and secure interfaces that **allow users to use one platform for all their DeFi needs.**

2) Set Owners, Managers Or Automation For DeFi Accounts



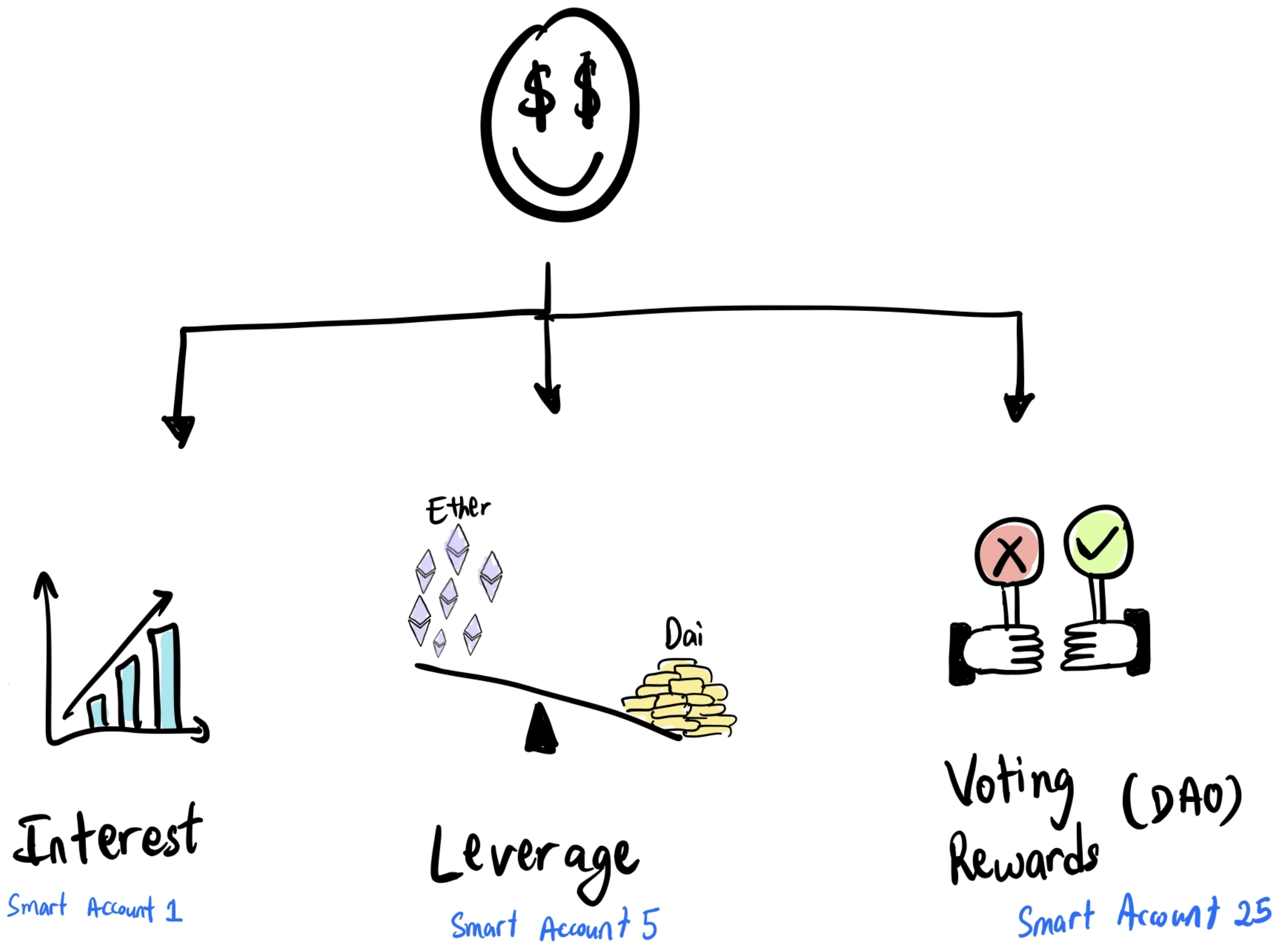
One of the biggest barriers to scaling DeFi for users is the difficulty for users to be able to set owners or delegate managers for specific assets or DeFi use cases according to the level of trustlessness users are comfortable with.

This is a critical step for DeFi usability since different users will require different levels of delegation for various use cases. In essence, some might want their private keys to certain assets to be stored elsewhere, some would want the optimization of assets to be trustlessly managed by an expert, and others might want to delegate the staking and voting for rewards.

In the new Instadapp, users will initially be able to set multiple owners for individual DeFi accounts. Moving forward, you can also add Instadapp as guardian or manager of your account to help w account recovery, fund management, etc.

Moving forward, developers will be able to set granular permissions for delegation. These means, building on DeFi accounts, they will be able to execute specific sets of DeFi actions for users, but they will not be able to transfer out the assets unless the user provides those permissions as well. This type of granularity is of major importance towards **trustless delegation**.

3) Scale DeFi Manageability With Multiple DeFi Accounts



DeFi assets are fundamentally more complex than traditional token assets to use and far harder to manage and track in terms of optimization, returns, etc. By managing everything under one Ethereum account, it becomes impossible to manage over time.

Instadapp and other developers building on top of DeFi Accounts can allow users to create multiple accounts for different use cases they might have. This will mirror the current financial world, where users have different accounts for different usage and investment needs, often managed by completely different people.

Combined with the rest of the features above, this means that users will not just be able to manage and track funds separately but also use different protocols, run analytics, or extend with shared ownership and delegation on each separate account.

For DeFi Developers

Ease of Composability

DeFi wizards can build complex cross-protocol transactions for DeFi Account holders without needing to write any solidity code. Here are some use cases that would require no or minimal, smart contract experience.

- Instadapp's protocol bridge to port assets between Maker and Compound, or any between other lending protocols.
- DeFiZap's composed transactions to deploy across multiple liquidity and lending protocols.
- iEarn's optimizer that automatically maximizes yield across protocols.
- DefiSaver's automated CDP saver to save positions when prices fluctuate.

As we add new connectors for new protocols, dApp developers will be able to compose their services with them without any need for new smart contracts or get their users to perform complex transactions.

Ease of Innovation

With built-in composability, liquidity bridges, and unlimited options for authorization, developer building on top of DeFi accounts will be able to easily innovate on new business and product models with minimal security worries.

- New Business Models: Build new revenue streams and networks from automating or helping users delegate their DeFi activities. Easily include their own set of fees to be added in the same transaction.
- Portability of Funds: With the existence of our liquidity bridge, their users will be able to easily port their positions or perform complex transactions between protocols using the connectors and liquidity pool.
- Built-in Security: Smart account transactions are secure by default, since no new smart contracts are deployed, which vastly simplifies and removes attack vectors.

These features are all designed to make it easy for developers to build products and use cases that advance the future of programmatic and networked money management.

For Wallets

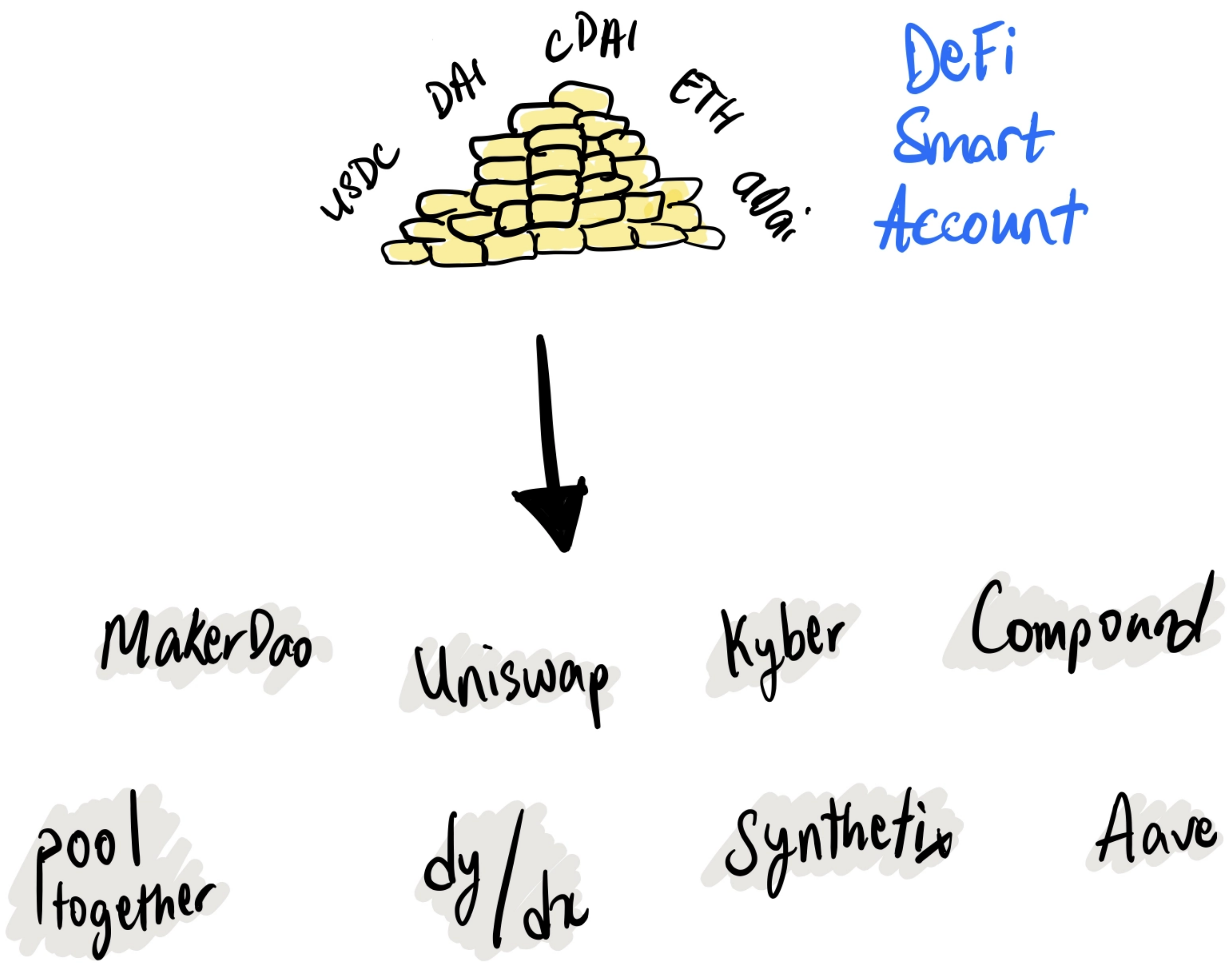
Wallets are typically the first and most trusted access points for users, which means it's the perfect place to introduce DeFi services. DeFi Accounts makes it easy to both build a DeFi "section" to complement the regular interface and innovate on new product and business models.

Think of the regular Ethereum account as the checking account, and DeFi accounts as the savings and investment accounts for the users, which you will be able to both offer new seamless use cases easily while being able to generate fees by performing key actions on behalf of the users.

Implementing extensible DeFi in your wallet becomes very simple since it creates a clear space for your users to easily store their DeFi assets and experiment with new use cases while remaining completely trustless and without interfering with their regular Ethereum account.

Importantly, using DeFi accounts also means that you are never locked into using any given protocols, and will be able to easily adapt your product based on the latest innovations or your own business needs.

For DeFi Protocols And Services



We are committed to helping drive awareness and adoption for new innovative protocols and services.

We would like to collaborate with all protocol developers to create connectors to extend the use cases for both Instadapp users as well as DeFi account holders. As discussed earlier, once connectors are created, developers will be able to easily offer the protocols as part of their interface or composed transactions.

We are also designing the Instadapp UX to encourage users to try out innovative new use cases. When they create a new Defi account, they will be encouraged to use the account to experiment with new DeFi services and protocols.

For Instadapp Users

We will be launching a new Instadapp alongside DeFi Accounts. The initial version of Instadapp will start simple and be built together with our users and the broader DeFi community. For current users of Instadapp, there will be a migration tool for migrating their assets to DeFi accounts.

The trustless nature of Instadapp (and DeFi accounts) means the **user's assets will always belong to the user, and they can easily try out in the new system without any rush to move their assets over until they are 100% confident**. There is no need to migrate their assets until they want to; the current Instadapp will remain fully functional.

We are currently brainstorming on exciting new use cases that would not have been possible using the current Instadapp architecture, including optimization across protocols, delegated voting, and a lot more - and would love to hear from users.

Talk to us!

Apart from launching the new Instadapp and DeFi Accounts, we will also begin to incrementally roll out developer docs and examples, as well as developer-focused communication groups, as we seek to build the future of programmatic money management together as a community.

We invite everyone interested in this to join us on this journey to create the next generation of DeFi together:

- If you are a DeFi user, tell us what you think of the features and ideas above and how you would like to expand your usage of DeFi
- If you are a developer who wants to leverage DeFi to its full potential for your users and innovate on new product and business models
- If you are a protocol developer who wants to allow Instadapp users and web3 developers to adopt and innovate on your service.

It is very early days for both InstaDApp and DeFi, and we would like to work on it together with the rest of the ecosystem.

> Instadapp Users Group

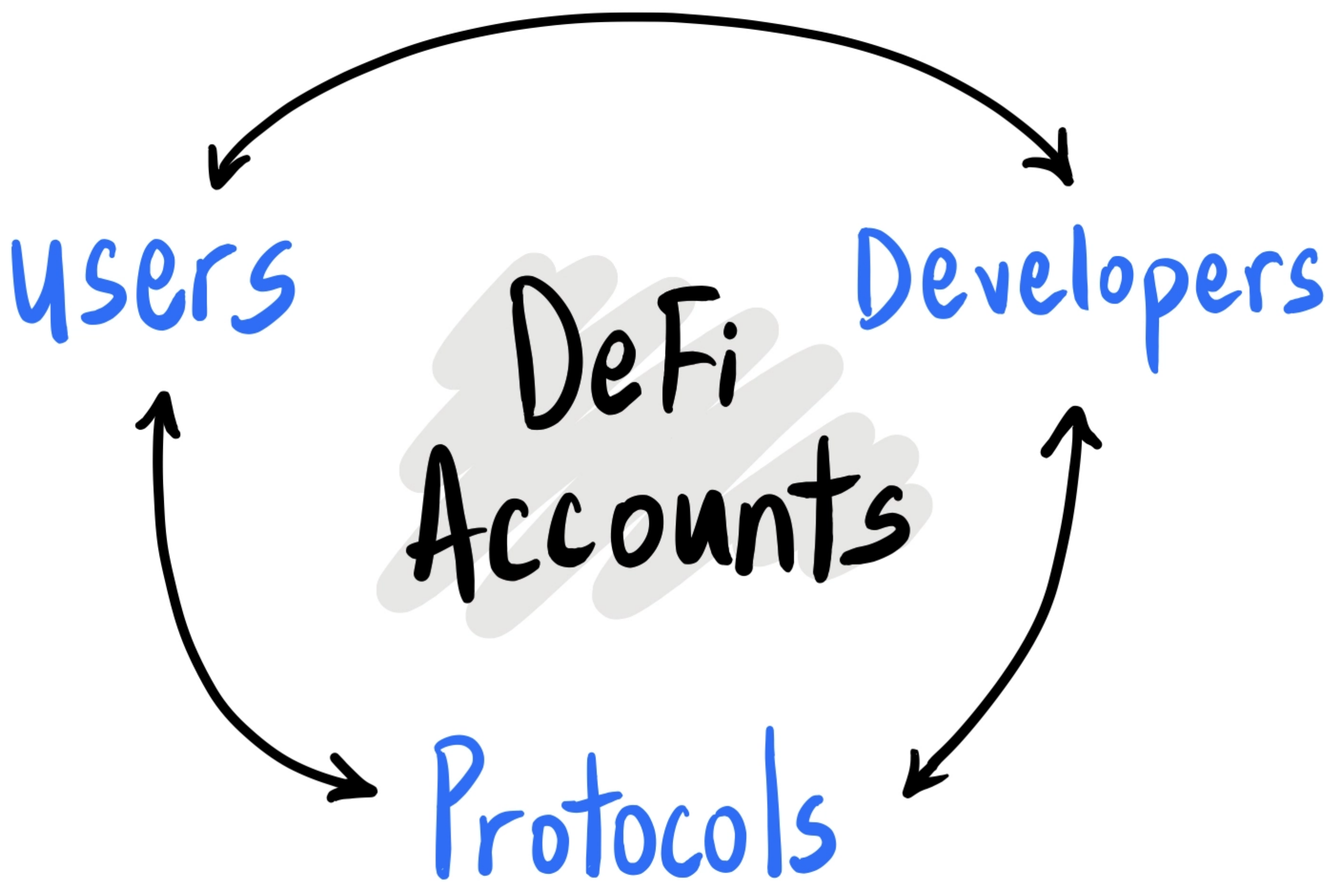
> Our Brand New DeFi Developers Group 😍

Summary

One of the main things we realized deeply is how improving the user experience of DeFi, far from just being an interface issue, involves deeply integrated efforts across the stack — from the underlying protocols to the smart contracts that glue it together, to the interfaces that the vast majority of users will see, to the myriad manners in delegated managers can take up key roles for the users.

For DeFi to go mainstream and remain open for usage and innovation, we have to improve the UX in DeFi along these key dimensions for both developers and users, along these key dimensions:

- Simplicity: Ability to manage and delegate their accounts seamlessly.
- Freedom: Ability to move, optimize, experiment with new use cases, and port assets between protocols seamlessly.
- Security: Ability for users to use DeFi securely according to their level of comfort, for most developers to be able to build complex dApps without smart contracts (and the security holes that might emerge), and allowing protocols to focus on their core layer.



This is an undertaking that has to be taken up as a cohesive effort by the ecosystem, and we are hopeful that DeFi Accounts can contribute our part to building the new foundation for the user experiences needed to make DeFi mainstream.

Litepaper

Version: 1.4 (March 2020)

Abstract

Synthetix is a decentralised synthetic asset issuance protocol built on Ethereum. These synthetic assets are collateralized by the Synthetix Network Token (SNX) which when locked in the contract enables the issuance of synthetic assets (Synths). This pooled collateral model enables users to perform conversions between Synths directly with the smart contract, avoiding the need for counterparties. This mechanism solves the liquidity and slippage issues experienced by DEX's. Synthetix currently supports synthetic fiat currencies, cryptocurrencies (long and short) and commodities. SNX holders are incentivised to stake their tokens as they are paid a pro-rata portion of the fees generated through activity on Synthetix.Exchange, based on their contribution to the network. It is the right to participate in the network and capture fees generated from Synth exchanges, from which the value of the SNX token is derived. Trading on Synthetix.Exchange does not require the trader to hold SNX.

SNX as collateral

How SNX backs Synths

All Synths are backed by SNX tokens. Synths are minted when SNX holders stake their SNX as collateral using Mintr, a decentralised application for interacting with the Synthetix contracts. Synths are currently backed by a 750% collateralisation ratio, although this may be raised or lowered in the future through community governance mechanisms. SNX stakers incur debt when they mint Synths, and to exit the system (i.e. unlock their SNX) they must pay back this debt by burning Synths.

Synthetix is also currently trialling Ether as an alternative form of collateral. This means traders can borrow Synths against their ETH and begin trading immediately, rather than needing to sell their ETH. Staking ETH requires a collateralisation ratio of 150% and creates a debt denominated in ETH, so ETH stakers mint sETH rather than sUSD and do not participate in the ‘pooled debt’ aspect of the system. In this model, ETH stakers do not receive fees or rewards as they take no risk for the debt pool.

Why SNX holders stake

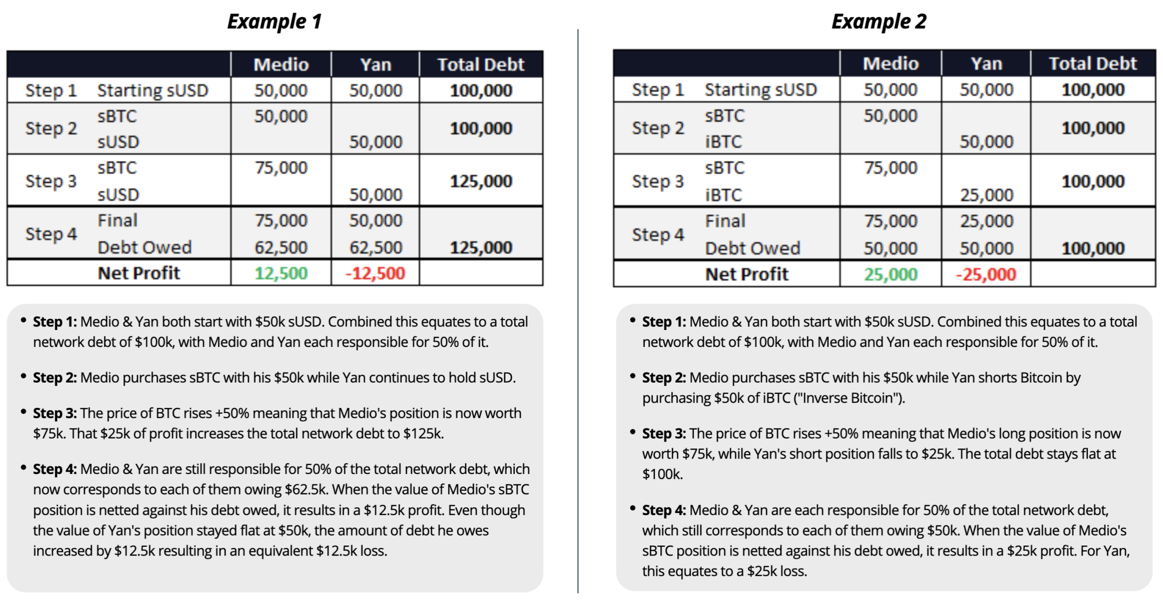
SNX holders are incentivised to stake their tokens and mint Synths in several ways. Firstly, there are exchange rewards. These are generated whenever someone exchanges one Synth to another (i.e. on Synthetix.Exchange). Each trade generates an exchange fee that is sent to a fee pool, available for SNX stakers to claim their proportion each week. This fee is between 10-100 bps (0.1% - 1%, though typically 0.3%), and will be displayed during any trade on Synthetix.Exchange. The other incentive for SNX holders to stake/mint is SNX staking rewards, which comes from the protocol’s inflationary monetary policy. From March 2019 to August 2023, the total SNX supply will increase from 100,000,000 to 260,263,816, with a weekly decay rate of 1.25% (from December 2019). From September 2023, there will be an annual 2.5% terminal inflation for perpetuity. These SNX tokens are distributed to SNX stakers weekly on a pro-rata basis provided their collateralisation ratio does not fall below the target threshold.

Minting, burning, and the C-Ratio

The mechanisms above ensure SNX stakers are incentivised to maintain their Collateralisation Ratio (C-Ratio) at the optimal rate (currently 750%). This ensures Synths are backed by sufficient collateral to absorb large price shocks. If the value of SNX or Synths fluctuate, each staker’s C Ratio will fluctuate. If it falls below 750% (although there is a small buffer allowing for minor fluctuations), they will be unable to claim fees until they restore their ratio. They adjust their ratio by either minting Synths if their ratio is above 750%, or burning Synths if their ratio is below 750%.

Stakers, debt, and pooled counterparties

SNX stakers incur a ‘debt’ when they mint Synths. This debt can increase or decrease independent of their original minted value, based on the exchange rates and supply of Synths within the network. For example, if 100% of the Synths in the system were synthetic Bitcoin (sBTC), which halved in price, the debt in the system would halve, and each staker’s debt would also halve. This means in another scenario, where only half the Synths across the system were sBTC, and BTC doubled in price, the system’s total debt—and each staker’s debt—would increase by one quarter. In this way, SNX stakers act as a pooled counterparty to all Synth exchanges; stakers take on the risk of the overall debt in the system. They have the option of hedging this risk by taking positions external to the system. By incurring this risk and enabling trading on Synthetix.Exchange stakers earn a right to fees generated by the system.



Example 1

		Medio	Yan	Total Debt
Step 1	Starting sUSD	50,000	50,000	**100,000**
Step 2	sBTC	50,000		**100,000**
	sUSD		50,000	
Step 3	sBTC	75,000		**125,000**
	sUSD		50,000	
Step 4	Final	75,000	50,000	
	Debt Owed	62,500	62,500	**125,000**
	Net Profit	12,500	-12,500	

- **Step 1:** Medio & Yan both start with $50k sUSD. Combined this equates to a total network debt of $100k, with Medio and Yan each responsible for 50% of it.
- **Step 2:** Medio purchases sBTC with his $50k while Yan continues to hold sUSD.
- **Step 3:** The price of BTC rises +50% meaning that Medio's position is now worth $75k. That $25k of profit increases the total network debt to $125k.
- **Step 4:** Medio & Yan are still responsible for 50% of the total network debt, which now corresponds to each of them owing $62.5k. When the value of Medio's sBTC position is netted against his debt owed, it results in a $12.5k profit. Even though the value of Yan's position stayed flat at $50k, the amount of debt he owes increased by $12.5k resulting in an equivalent $12.5k loss.

Example 2

		Medio	Yan	Total Debt
Step 1	Starting sUSD	50,000	50,000	**100,000**
Step 2	sBTC	50,000		**100,000**
	iBTC		50,000	
Step 3	sBTC	75,000		**100,000**
	iBTC		25,000	
Step 4	Final	75,000	25,000	
	Debt Owed	50,000	50,000	**100,000**
	Net Profit	25,000	-25,000	

- **Step 1:** Medio & Yan both start with $50k sUSD. Combined this equates to a total network debt of $100k, with Medio and Yan each responsible for 50% of it.
- **Step 2:** Medio purchases sBTC with his $50k while Yan shorts Bitcoin by purchasing $50k of iBTC ("Inverse Bitcoin").
- **Step 3:** The price of BTC rises +50% meaning that Medio's long position is now worth $75k, while Yan's short position falls to $25k. The total debt stays flat at $100k.
- **Step 4:** Medio & Yan are each responsible for 50% of the total network debt, which still corresponds to each of them owing $50k. When the value of Medio's sBTC position is netted against his debt owed, it results in a $25k profit. For Yan, this equates to a $25k loss.

Examples from Delphi Digital demonstrating how debt works in the Synthetix system.

Synth Pegging Mechanism

The Synth peg is critical to a well functioning system, because traders require both liquidity and stability between a Synth/s and other cryptoassets in order to take profits from trading. Some Synths trade on the open market, so it is possible for them to fall below par with the assets they track. Incentives are required to ensure that deviations from the peg are minimal and that actors are motivated to correct them.

There are three methods to maintain the Synth peg:

- **Arbitrage**: SNX stakers have created debt by minting Synths, so if the peg drops they can now profit by buying sUSD back below par and burning it to reduce their debt, as the Synthetix system always values 1 sUSD at $1 USD.
- **sETH liquidity pool on Uniswap**: each week, a portion of the SNX added to the total supply through the inflationary monetary policy is distributed as reward to people providing sETH/ETH liquidity on Uniswap. This has incentivised liquidity providers to collectively create the largest liquidity pool on Uniswap (at time of writing), allowing traders to purchase Synths to start trading or sell Synths to take profits.
- **SNX auction**: Synthetix is currently trialling a new mechanism with the dFusion protocol (from Gnosis) in which discounted SNX is sold at auction for ETH, which is then used to purchase Synths below the peg.

Synthetix.Exchange

Why trade synthetic assets?

Synthetic assets provide exposure to an asset without holding the underlying resource. This has a range of advantages, including reducing the friction when switching between different assets (e.g. from Apple shares to synthetic gold), expanding the accessibility of certain assets, and censorship resistance.

Advantages of Synthetix.Exchange

Trading on Synthetix.Exchange provides many advantages over centralised exchanges and order book based DEX’s. The lack of an order book means all trades are executed against the contract, known as P2C (peer-to-contract) trading. Assets are assigned an exchange rate through price feeds supplied by an oracle, and can be converted using the Synthetix.Exchange dApp. This provides infinite liquidity up to the total amount of collateral in the system, zero slippage, and permissionless on-chain trading.

How Synths work

Synths are synthetic assets that track the price of the underlying asset. They allow holders to gain exposure on Ethereum to various asset classes without holding the underlying assets themselves or trusting a custodian. Synths are backed by the Synthetix Network Token (SNX), which is staked as collateral at a ratio of 750%.

The current Synths

There are currently five categories of Synths available: fiat currencies, commodities, cryptocurrencies, inverse cryptocurrencies, and cryptocurrency indexes. Our fiat Synths include sUSD, sEUR, sKRW, and many more; our commodity Synths include synthetic gold and synthetic silver, both measured per ounce; our cryptocurrencies include sBTC, sETH, and sBNB, with more to come; and our Inverse Synths inversely track the price of those available cryptocurrencies, meaning that when BTC's price decreases, iBTC's price increases. Our current cryptocurrency indexes are sDEFI and sCEX (and their inverses), which respectively track a basket of DeFi assets and a basket of centralised exchange tokens.

System Architecture

Minting Synths

An SNX holder can mint sUSD by locking their SNX as collateral via the Synthetix smart contract. The steps involved when an SNX holder mints are:

- The Synthetix contract checks that the SNX staker can mint Synths against their SNX, which requires their Collateralisation Ratio to be below 750%.

- Their debt is added to the Debt Register. The debt is the amount of the new value minted, and is stored in sUSD
- With the debt assigned to the staker, the Synthetix contract instructs the sUSD contract to issue the new amount. It adds it to its total supply and assigns the newly minted sUSD to the user's wallet.

If the price of SNX increases, an equivalent portion of a staker's SNX is automatically unlocked as collateral. For example, if a user locks $100 of SNX as collateral, and the value of SNX doubles, then half of their SNX (total value: $200) is locked and the other half is unlocked. If they wish, that extra unlocked SNX can then be staked to mint more sUSD.

Exchanges

The steps involved for the smart contracts to process a Synth exchange (from sUSD to sBTC in this example) are below:

- Burn the source Synth (sUSD), which involves reducing that wallet address's sUSD balance and updating the total supply of sUSD.
- Establish the conversion amount (i.e. the exchange rate, based on the price of each currency).
- Charge an exchange fee, which is currently 0.3% of the converted amount, and send the fee as sUSD to the fee pool, where it can be claimed by SNX stakers.
- The remaining 99.7% is issued by the destination Synth (sBTC) contract and the wallet address balance is updated
- The sBTC total supply is updated.

No counterparty is required to exchange, as the system converts the debt from one Synth to another. Hence no order books or order matching is required, resulting in infinite liquidity between Synths. No debt change is required to be recorded against the debt pool either, as the same value is burned from the source Synth and minted from the destination Synth.

Claiming Fees

When Synths are exchanged through the Synthetix contract, a 0.3% fee is extracted and sent to the fee pool to be claimed by SNX stakers. When claiming fees (also called Synth exchange rewards) a staker also claims their SNX staking rewards, which reward them with extra SNX for staking the SNX they currently have. The smart contracts' process once a staker requests to claim their fees is as follows:

- The fee pool checks whether there are fees currently available and whether the staker is eligible to receive fees.
- The amount of fees in sUSD is sent to the staker's wallet address and the balance of the fee pool is updated.
- Additionally, a pro-rata amount of escrowed SNX is assigned to the wallet address from the SNX staking rewards contract.

Fees are allocated based on the proportion of debt each staker has issued. For example, if a staker has issued 1,000 sUSD in debt, the debt pool is 10,000 sUSD, and 100 in fees are generated in a fee period, this staker is entitled to 10 sUSD because their debt represents 10% of the debt pool. The same proportional distribution mechanism is used for SNX staking rewards.

Burning debt

When an SNX staker wants to exit the system or reduce their debt and unlock staked SNX, they must pay back their debt. At its simplest: a staker mints 10 sUSD by locking SNX as collateral, and must burn 10 sUSD to unlock it. But if the debt pool fluctuates (and therefore their individual debt fluctuates) while they are staked, they may need to burn more or less debt than they minted. The process for reducing debt to zero is as follows:

- The Synthetix contract determines their debt balance and removes them from the Debt Register.
- The required amount of sUSD is burned, and total supply of sUSD is updated along with the sUSD balance in the user's wallet.
- Their SNX balance becomes transferrable.

The debt pool

The system tracks the debt pool (as well as each individual staker's debt) each time an SNX holder mints or burns Synths. It does this by updating the Cumulative Debt Delta Ratio. This measures the SNX staker's proportion of the debt pool at the time they last minted or burned, as well as the debt change caused by other stakers entering or leaving the system. The system uses this information to determine the individual debt of each staker at any time in the future, without having to actually record the changing debt of each individual staker.

Updating the Cumulative Debt Delta Ratio on the Debt Register allows the system to track every user's % of the debt. It calculates the % change the new debt introduces against the debt pool using the formula below and appends it to the Debt Register:

`New Debt Minted ( Total Existing Debt + New Debt)`

The staker's last mint/burn action is then recorded in the Debt Register within their issuance data and the relative index number at which this action happened. The detail recorded is the percentage of the debt pool they represent, which is calculated by this formula:

`User debt percentage =(New Debt + Existing Debt) (Previous Debt Pool + New Debt)`

The Debt Register holds the Cumulative Debt Delta Ratio, which is the product of the calculation above, and the relative time (index) the debt was added, so that it can be used to calculate any user's % of the debt pool at any index in the future based on the % shift in the debt pool their last mint/burn caused.

We recalculate the debt pool by summing the number of tokens in each Synth contract multiplied by the current exchange rates, each time new debt is issued/burned:

`totalDebtIssued = totalIssuedSynths`

This enables the calculation of the current debt pool, and is included in the updated Cumulative Debt Delta Ratio so that we know at each Debt Register entry the size of the debt (in Synths).

When a staker pays back their debt (i.e. by burning the Synths they minted) to unlock their SNX collateral the system updates the Cumulative Debt Delta based on the % shift in the amount of debt to be burned against the total value of the system's debt after the reduction in debt.

This is the inverse calculation from when a user mints new debt:

`user's new debt percentage =(existing debt - debt to be burned) (debt pool - debt to be burned)`

This is the formula for calculating the updated Cumulative Debt Delta:

`delta = debt to be burned (debt pool -debt to be burned)`

If a staker burns all their debt, their issuance data in the Debt Register will be set to 0 and they will no longer be part of the debt pool.

The oracle

The value of all synthetic assets in the Synthetix system are currently determined by oracles that push price feeds on-chain. It uses an algorithm with a variety of sources to form an aggregate value for each asset. The price feeds are currently supplied by both Chainlink's independent node operators and Synthetix, and will soon all be supplied by Chainlink.

Current Risks and Risk Mitigation Strategies

Current risks

There are several risks in the current architecture, as Synthetix is still an experimental system and complex systems require both empirical observations and theoretical analysis. Empirical observation and theoretical analysis ensure the mechanism design aligns incentives for all players.

One risk involves the debt SNX holders issue when they stake their SNX and mint Synths. As previously explained, this debt can fluctuate due to exchange rate shifts within the system. This means that to exit the system and unlock their staked SNX, they may need to burn more Synths than they originally minted.

Most people in the cryptocurrency space are aware of this risk, but the prices of most cryptoassets are highly correlated to Bitcoin and/or Ethereum. This means it’s possible for major price fluctuations in the SNX token to occur for reasons that have little to do with SNX or the Synthetix system.

Finally, there are a number of aspects of the system that are currently centralised. This decision has been made to ensure efficient implementation of the project. One example of centralisation is the use of proxy contracts across much of the architecture. This is to ensure the system can be upgraded easily but confers a level of control to the engineering team which requires trust from users. While these aspects will be phased out over time, it is important to understand the risks inherent in the current system architecture.

Risk mitigation strategies

As a decentralised protocol, the Synthetix team is committed to decentralisation and censorship resistance — this will be a gradual process as the system matures. This includes crucial areas such as our price feeds. We have previously announced a partnership with Chainlink, a provider of decentralised oracle solutions.

Another important area is governance, we have recently initiated regular community governance calls to ensure the project’s goals are aligned with the community. Another aspect of this process is a move to a formal change management process, we have introduced SIP’s (Synthetix Improvement Proposals) to allow the community to introduce change requests and to ensure that any changes to the system are well understood and considered by all stakeholders.

Future Functionality

Additional Synths

There are many different kinds of Synths that can be added to the system to provide greater utility to Synthetix.Exchange. These include leveraged assets that are not available on other platforms as well as indices like the S&P500 and equities like APPL and TSLA.

Synthetic futures

We expect to launch the ability for traders to take synthetic futures on Synthetix.Exchange in the near future. Many aspects of this functionality are yet to be finalised, but it's expected it will use a self balancing mechanism similar to the Uniswap auto market maker algorithm, where the total open interest of each position and therefore the risk to SNX stakers is capped and borrow rates are adjusted based on the current open interest. The system will also encourage traders to balance the risk in the system by paying a percentage of the fees to traders who rebalance positions, though this feature will not be in the initial release. There are already a number of derivatives trading platforms for cryptoassets, but they are all limited by counterparty liquidity. The unique design of the Synthetix system means it may be able to capture market share in this area, similarly to how Binance captured market share by listing more cryptoassets than most other centralised exchanges.

Leveraged trading

Leveraged trading drives a significant amount of volume on crypto exchanges, and while synthetic futures will compete directly with centralised futures platforms, there is a lot of value in supporting tokenised leverage.

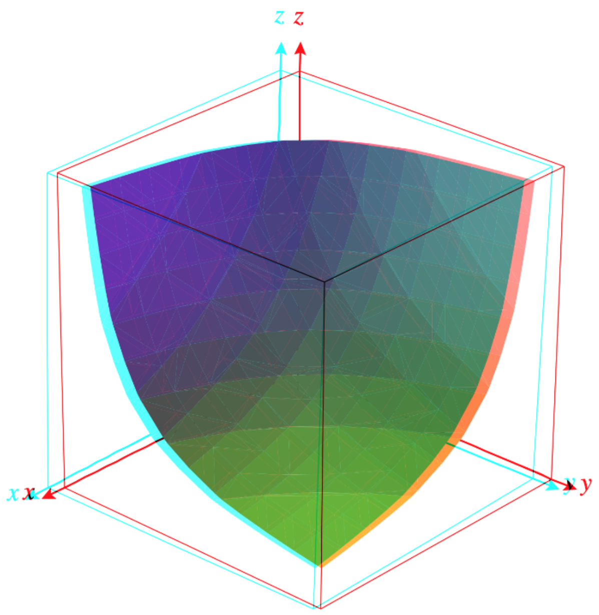
Advanced order types

The current version of Synthetix.Exchange supports only market orders which limits the usability of the exchange. An advanced order engine will be able to support limit, stop loss, stop limits, and other advanced order types. This will use a relay network for processing advanced orders. Advanced order types are critical to reaching feature parity with centralised exchanges.

Conclusion

Synthetix has already delivered one of the most complex and useful protocols built on Ethereum to date. But the potential for censorship-resistant synthetic assets is still largely untapped. Further improvements to the mechanism as well as functional upgrades and new Synths will vastly increase the utility of the platform. Movement to a decentralised governance process will also reduce systemic risk and increase the long term viability of the project.

Whitepaper



A non-custodial portfolio manager, liquidity provider, and price sensor.

by:

Fernando Martinelli

Nikolai Mushegian

v2019-09-19

contact@balancer.finance (mailto:contact@balancer.finance)

Introduction

A Balancer Pool is an automated market maker with certain key properties that cause it to function as a *self-balancing weighted portfolio* and *price sensor*.

Balancer turns the concept of an index fund on its head: instead of paying fees to portfolio managers to rebalance your portfolio, you collect fees from traders, who rebalance your portfolio by following arbitrage opportunities.

Balancer is based on a particular N-dimensional surface which defines a cost function for the exchange of any pair of tokens held in a Balancer Pool. This approach was first described by V. Buterin[0] (https://www.reddit.com/r/ethereum/comments/55m04x/lets_run_onchain_decentralized_exchanges_the_way/), generalized by Alan Lu[1] (https://blog.gnosis.pm/building-a-decentralized-exchange-in-ethereum-eea4e7452d6e), and proven viable for market making by the popular Uniswap[2] (https://uniswap.io) dapp.

We independently arrived at the same surface definition by starting with the requirement that any trade must maintain a constant proportion of value in each asset of the portfolio. We applied an invariant-based modeling approach described by Zargham et al[3] (https://arxiv.org/pdf/1807.00955.pdf) to construct this solution. We will prove that these constant-value market makers have this property.

Table of Contents

Present Work

Index funds are a common financial instrument. The first index fund became effective in 1972. Ever since, investors rely heavily on different portfolio strategies to hedge risk and achieve diversification. Index funds guarantee investors a constant and controlled exposure to a portfolio. If one of its assets out- or under-performs, it is respectively sold or bought to keep its value share of the total portfolio constant.

Both in the conventional financial system as well as in the blockchain context, index funds and other types of investment portfolios charge investors fees for managing and holding their funds. These fees are necessary to pay for the costs of actively rebalancing the index funds, be it by manual traders or automatic bots.

There are many centralized solutions for portfolio management and for investing in index funds. These all share some form of custodial risk.

We are aware of one decentralized (read: non-custodial) solution that shares all the fundamental characteristics Balancer was designed to have: Uniswap (https://uniswap.io). This approach was first described by V. Buterin (https://www.reddit.com/r/ethereum/comments/55m04x/lets_run_onchain_decentralized_exchanges_the_way/) and generalized by Alan Lu (https://blog.gnosis.pm/building-a-decentralized-exchange-in-ethereum-eea4e7452d6e).

We independently arrived at the same surface definition by *starting* with the requirement that any trade must maintain a constant proportion of value in each asset of the portfolio. We applied an invariant-based modeling approach described by Zargham et al (https://arxiv.org/pdf/1807.00955.pdf) to construct this solution. We will prove that these constant-value market makers have this property.

Theory

Throughout this paper, we use the term "token" to refer to a generic asset because our first implementation is a contract system that manipulates ERC20 tokens on the Ethereum network. However, there is nothing fundamental about the Ethereum execution context that enables this market-making algorithm, which could be offered by a traditional financial institution as a centralized (custodial) product.

Value Function

The bedrock of Balancer's exchange functions is a surface defined by constraining a value function V — a function of the pool's weights and balances — to a constant. We will prove that this surface implies a spot price at each point such that, no matter what exchanges are carried out, the share of value of each token in the pool remains constant.

The value function V is defined as:

$$V = \prod_t B_t^{W_t} \tag{1}$$

Where

- t ranges over the tokens in the pool;
- B_t is the balance of the token in the pool;
- W_t is the normalized weight of the token, such that the sum of all normalized weights is 1.

By making V constant we can define an invariant-value surface as illustrated in Fig.0.

Spot Price

Each pair of tokens in a pool has a spot price defined entirely by the weights and balances of just that pair of tokens. The spot price between any two tokens, $SpotPrice_i^o$, or in short SP_i^o, is the the ratio of the token balances normalized by their weights:

$$SP_i^o = \frac{\frac{B_i}{W_i}}{\frac{B_o}{W_o}} \tag{2}$$

Where:

- B_i is the balance of token *i*, the token being sold by the trader which is going *into* the pool.
- B_o is the balance of token *o*, the token being bought by the trader which is going *out* of the pool.
- W_i is the weight of token *i*
- W_o is the weight of token *o*

From this definition it is easy to see that if weights are held constant, the spot prices offered by Balancer Pools only change with changing token balances. If the pool owner does not add or remove tokens to/from the pool, token balances can only change through trades. The constant surface causes the price of tokens being bought by the trader (token *o*) to increase and price of tokens being sold by the trader (token i) to decrease. One can prove that whenever external market prices are different from those offered by a Balancer Pool, an arbitrageur will make the most profit by trading with that pool until its prices equal those on the external market. When this happens there is no more arbitrage opportunity. These arbitrage opportunities guarantee that, in a rational market, prices offered by any Balancer Pool move in lockstep with the rest of the market.

Effective Price

It is important to bear in mind that SP_i^o is the *spot* price, which is the theoretical price for infinitesimal trades, which would incur no slippage. In reality, the effective price for any trade depends on the amount being traded, which always causes a price change. If we define A_o as the amount of token o being bought by the trader and A_i as the amount of token i being sold by the trader, then we can define the Effective Price as:

$$EP_i^o = \frac{A_i}{A_o} \tag{3}$$

And as mentioned above, EP tends to SP when traded amounts tend to 0:

$$SP_i^o = \lim_{A_o, A_i \to 0} EP_i^o \tag{4}$$

Spot Price Proof

Let's now prove that this choice of V entails Eq.2.

First of all, we know that what the trader buys, A_o, is subtracted from the contract's balance. Therefore $A_o = -\Delta B_o$. Likewise, what the trader sells, A_i, is added to the contract's balance. Therefore $A_i = \Delta B_i$.

Substituting in Eq.2 and Eq.3 we get:

$$SP_i^o = \lim_{A_o, A_i \to 0} EP_i^o = \lim_{\Delta B_o, \Delta B_i \to 0} \frac{\Delta B_i}{-\Delta B_o} \tag{5}$$

This limit is, by definition, minus the partial derivative of B_i in function of B_o:

$$SP_i^o = -\frac{\partial B_i}{\partial B_o} \tag{6}$$

From the value function definition in Eq.1 we can isolate B_i:

$$B_i^{W_i} = \frac{V}{\left(\prod_{k \neq i,o} B_k^{W_k}\right) . B_o^{W_o}}$$

$$B_i = \left(\frac{V}{\left(\prod_{k \neq i,o} B_k^{W_k}\right) . B_o^{W_o}}\right)^{\frac{1}{W_i}} \tag{7}$$

Now we use Eq.7 to expand the partial derivative in Eq.6:

$$SP_i^o = -\frac{\partial B_i}{\partial B_o} = -\frac{\partial}{\partial B_o}\left(\left(\frac{V}{\left(\prod_{k \neq i,o}(B_k)^{W_k}\right) \cdot (B_o)^{W_o}}\right)^{\frac{1}{W_i}}\right) =$$

$$-\left(\frac{V}{\prod_{k \neq i,o}(B_k)^{W_k}}\right)^{\frac{1}{W_i}} \cdot \frac{\partial}{\partial B_o}\left(B_o^{-\frac{W_o}{W_i}}\right) =$$

$$-\left(\frac{V}{\prod_{k \neq i,o}(B_k)^{W_k}}\right)^{\frac{1}{W_i}} \cdot -\frac{W_o}{W_i} \cdot B_o^{-\frac{W_o}{W_i}-1} =$$

$$\left(\frac{V}{\prod_k (B_k)^{W_k}}\right)^{\frac{1}{W_i}} \cdot B_o^{\frac{W_o}{W_i}} \cdot B_i \cdot \frac{W_o}{W_i} \cdot B_o^{-\frac{W_o}{W_i}-1} =$$

$$\left(\frac{V}{V}\right)^{\frac{1}{W_i}} \cdot B_o^{\frac{W_o}{W_i}} \cdot B_o^{-\frac{W_o}{W_i}} \cdot \frac{B_i}{W_i} \cdot \frac{W_o}{B_o} = \frac{\frac{B_i}{W_i}}{\frac{B_o}{W_o}}$$

which concludes our proof.

Constant Value Distribution Proof

We will now prove that:

1. Balancer Pools maintain a constant share of value across all tokens in the pool and;
2. These shares of value are equal to the weights associated to each token.

Let's calculate V^t, the total pool value in terms of an arbitrary token t from the pool. Since we already know that the pool has B_t tokens t, let's calculate how many tokens t all the other remaining tokens are worth. It does not make sense to use their Effective Price relative to token t since we are not going to do any actual trade. Instead, to calculate the theoretical value we use their Spot Price relative to token t.

From Eq.2 we can calculate V_n^t, i.e how many tokens t the balance of each token n is worth:

$$V_n^t = \frac{B_n}{SP_n^t} = B_n \cdot \frac{\frac{B_t}{W_t}}{\frac{B_n}{W_n}} = B_t \cdot \frac{W_n}{W_t} \tag{8}$$

We know that the total pool value in terms of tokens t is the sum of the values of each token in terms of tokens t:

$$V^t = \sum_k V_k^t = B_t + \sum_{k \neq t} V_k^t = B_t + \frac{B_t}{W_t} \cdot \sum_{k \neq t} W_n = \frac{B_t}{W_t} \cdot (W_t + \sum_{k \neq t} W_n) = \frac{B_t}{W_t} \tag{9}$$

Now to calculate S_n, the share of value each token n represents in the pool, all we have to do is divide the value of each token n, V_n^t, by the total pool value, V^t:

$$S_n = \frac{V_n^t}{V^t} = W_n \tag{10}$$

which proves both that the share each token represents of the total pool value is constant and also that it is equal to the weight of that token.

Trading Formulas

Calculating the trade outcomes for any given Balancer Pool is easy if we consider that the Value Function must remain invariant, i.e. V must have the same value before and after any trade.
In reality, V will increase as a result of trading fees applied after a trade state transition.
For more details on fees, see Implementation: Swap and Exit Fees

Out-Given-In

When a user sends tokens i to get tokens o, all other token balances remain the same. Therefore, if we define A_i and A_o as the amount of tokens i and o exchanged, we can calculate the amount A_o a users gets when sending A_i. Knowing the value function after the trade should be the same as before the trade, we can write:

$$\prod_{k \neq i,o} (B_k)^{W_k} \cdot (B_o - A_o)^{W_o} \cdot (B_i + A_i)^{W_i} = \prod_k (B_k)^{W_k} \tag{11}$$

$$\prod_{k \neq i,o} (B_k)^{W_k} \cdot (B_o - A_o)^{W_o} \cdot (B_i + A_i)^{W_i} = \prod_{k \neq i,o} (B_k)^{W_k} \cdot B_o^{W_o} \cdot B_i^{W_i} \tag{12}$$

$$(B_o - A_o)^{W_o} \cdot (B_i + A_i)^{W_i} = B_o^{W_o} \cdot B_i^{W_i} \tag{13}$$

$$B_o - A_o = \frac{B_i^{\frac{W_i}{W_o}} \cdot B_o}{(B_i + A_i)^{\frac{W_i}{W_o}}} \tag{14}$$

$$A_o = B_o \cdot \left(1 - \left(\frac{B_i}{B_i + A_i}\right)^{\frac{W_i}{W_o}}\right) \tag{15}$$

In-Given-Out

It is also very useful for traders to know how much they need to send of the input token A_i to get a desired amount of output token A_o. We can calculate the amount A_i as a function of A_o similarly as follows:

$$\prod_{k \neq i,o} (B_k)^{W_k} \cdot (B_o - A_o)^{W_o} \cdot (B_i + A_i)^{W_i} = \prod_k (B_k)^{W_k} \tag{16}$$

$$\prod_{k \neq i,o} (B_k)^{W_k} \cdot (B_o - A_o)^{W_o} \cdot (B_i + A_i)^{W_i} = \prod_{k \neq i,o} (B_k)^{W_k} \cdot B_o^{W_o} \cdot B_i^{W_i} \tag{17}$$

$$(B_o - A_o)^{W_o} \cdot (B_i + A_i)^{W_i} = B_o^{W_o} \cdot B_i^{W_i} \tag{18}$$

$$B_i + A_i = \frac{B_o^{\frac{W_o}{W_i}} \cdot B_i}{(B_o - A_o)^{\frac{W_o}{W_i}}} \tag{19}$$

$$A_i = B_i \cdot \left(\left(\frac{B_o}{B_o - A_o}\right)^{\frac{W_o}{W_i}} - 1\right) \tag{20}$$

Notice that A_o as defined by Eq.11 tends to $SP_i^o \cdot A_i$ when $A_i << B_i$, as expected. This can be proved by using L'Hopital's rule, but this proof is out of the scope of this paper.

In-Given-Price

For practical purposes, traders intending to use our contract for arbitrage will like to know what amount of tokens i – A_i – they will have to send to the contract to change the current spot price SP_i^o to another desired one $SP_i'^o$. The desired spot price will usually be the external market price and, so long as the contract spot price differs from that of the external market, any arbitrageur can profit by trading with the contract and bringing the contract price closer to that of the external market.

The highest profit possible by an arbitrageur is when they bring the contract spot price exactly to that of the external market. As already mentioned, this is the main reason why our design is successful in keeping track of the market prices. This makes it a reliable on-chain price sensor when implemented on a blockchain.

It can be proven that the amount of tokens i – A_i – a user needs to trade against tokens o so that the pool's spot price changes from SP_i^o to $SP_i'^o$ is:

$$A_i = B_i \cdot \left(\left(\frac{SP_i'^o}{SP_i^o} \right)^{\left(\frac{W_o}{W_o+W_i}\right)} - 1 \right) \tag{21}$$

Liquidity Providing Formulas

Pool Tokens

Pools can aggregate the liquidity provided by several different users. In order for them to be able to freely deposit and withdraw assets from the pool, Balancer Protocol has the concept of pool tokens. Pool tokens represent ownership of the assets contained in the pool. The outstanding supply of pool tokens is directly proportional to the Value Function of the pool. If a deposit of assets increases the pool Value Function by 10%, then the outstanding supply of pool tokens also increases by 10%. This happens because the depositor is issued 10% of new pool tokens in return for the deposit.

There are two ways in which one can deposit assets to the pool in return for pool tokens or redeem pool tokens in return for pool assets:

- Weighted-asset deposit/withdrawal
- Single-asset deposit/withdrawal

All-Asset Deposit/Withdrawal

An “all-asset” deposit has to follow the distribution of existing assets in the pool. If the deposit contains 10% of each of the assets already in the pool, then the Value Function will increase by 10% and the depositor will be minted 10% of the current outstanding pool token supply. So to receive P_{issued} pool tokens given an existing total supply of P_{supply}, one needs to deposit D_k tokens k for each of the tokens in the pool:

$$D_k = \left(\frac{P_{supply} + P_{issued}}{P_{supply}} - 1 \right) \cdot B_k \tag{22}$$

Where B_k is the token balance of token k before the deposit.

Similarly, a weighted-asset withdrawal is the reverse operation where a pool token holder redeems their pool tokens in return for a proportional share of each of the assets held by the pool. By redeeming $P_{redeemed}$ pool tokens given an existing total supply of P_{supply}, one withdraws from the pool an amount A_k of token k for each of the tokens in the pool:

$$A_k = \left(1 - \frac{P_{supply} - P_{redeemed}}{P_{supply}}\right) \cdot B_k \tag{23}$$

Where B_k is the token balance of token k before the withdrawal.

Single-Asset Deposit/Withdrawal

When a user wants to provide liquidity to a pool because they find its distribution of assets interesting, they may likely not have all of the assets in the right proportions required for a weighted-asset deposit.

Balancer allows anyone to get pool tokens from a shared pool by depositing a single asset to it, provided that the pool contains that asset.
Depositing a single asset A to a shared pool is equivalent to depositing all pool assets proportionally and then selling more of asset A to get back all the other tokens deposited. This way a depositor would end up spending only asset A, since the amounts of the other tokens deposited would be returned through the trades.

The amount of pool tokens one gets for depositing a single asset to a shared pool can be derived from the Value Function described above.

Single-Asset Deposit

The increase in the pool token supply proportional to the increase in the Value Function. If we define P_{issued} as the amount of pool tokens issued in return for the deposit, then:

$$\frac{V'}{V} = \frac{P'_{supply}}{P_{supply}} = \frac{P_{supply} + P_{issued}}{P_{supply}}$$

$$P_{issued} = P_{supply} \cdot \left(\frac{V'}{V} - 1\right) \tag{24}$$

Where V' is the Value Function after the deposit and V is the Value Function before the deposit.
Considering also B'_k the balance of asset k after the deposit and B_k its balance before the deposit, we have:

$$\frac{V'}{V} = \frac{\prod_k (B'_k)^{W_k}}{\prod_k (B_k)^{W_k}}$$

Let’s say the single-asset deposit was done in asset t, then the balances of all other tokens do not change after the deposit. We can then write:

$$\frac{V'}{V} = \frac{\prod_k (B'_k)^{W_k}}{\prod_k (B_k)^{W_k}} = \frac{(B'_t)^{W_t}}{(B_t)^{W_t}} = \left(\frac{B'_t}{B_t}\right)^{W_t}$$

If we define A_t as the amount deposited in asset t, then the new pool balance of asset t is $$B’*t = B_t + A_t$$. We can then substitute and get the final formula for the amount of new pool tokens issued $P*{issued} $inreturn for asinge - assetdepositI_t:

$$P_{issued} = P_{supply} \cdot \left(\left(1 + \frac{A_t}{B_t} \right)^{W_t} - 1 \right) \tag{25}$$

Single-Asset Withdrawal

When a pool token holder wants to redeem their pool tokens $P_{redeemed}$ in return for a single asset t, the amount withdrawn in asset t, A_t, is:

$$A_t = B_t \cdot \left(1 - \left(1 - \frac{P_{redeemed}}{P_{supply}} \right)^{\frac{1}{W_t}} \right) \tag{26}$$

Where B_t is the pool balance of asset t before the withdrawal.

Indeed, using the formulas of deposit and withdrawal defined above, not considering any fees, if one deposits A_t asset t for P_{issued} pool tokens and then redeems that same amount of pool tokens for asset t, they will get the same initial A_t back.

Trading Fees for Single-Asset Deposit Withdrawal

Depositing or withdrawing to/from a shared pool in a single asset t is equivalent to trading $(1 - W_t)$ of the amount deposited for all the other assets in the pool. W_t of the amount deposited is held by the pool already in the form of asset t, so charging a trading fee on that share would be unfair.

Indeed, if we disregard any possible pool exit fees, depositing only asset i and instantly withdrawing asset o will incur in the same trading fees as doing the trade from i to o using the trade function the pool offers.

Implementation

There are a few initial notes regarding the first release of Balancer. We will release a much more detailed explanation of the system at the same time that the source code is released.

Free Software on Ethereum

Balancer is implemented as a GPL3-licensed Ethereum smart contract system.

Releases

The 🍂**Bronze Release**🍂 is the first of 3 planned releases of the Balancer Protocol. Bronze emphasizes code clarity for audit and verification, and does not go to great lengths to optimize for gas.

The ❄️**Silver Release**❄️ will bring many gas optimizations and architecture changes that will reduce transaction overhead and enable more flexibility for controlled pools.

The ☀️**Golden Release**☀️ will introduce several new features to tie the whole system together.

Numerical Algorithms

The formulas in the Theory section are sufficient to describe the functional specification, but they are not straightforward to implement for the EVM, in part due to a lack of mature fixed-point math libraries.

Our implementation uses a combination of a few algebraic transformations, approximation functions, and numerical hacks to compute these formulas with bounded maximum error and reasonable gas cost.

The rest of this section will be released at the same time as the Bronze release source code.

Controlled vs Finalized Pools

The 🍂Bronze Release🍂 allows two basic tiers of trust with respect to pools:

1. *Controlled* pools are configurable by a “controller” address. Only this address can add or remove liquidity to the pool (call `join` or `exit`). This type of pool allows the change of pool assets types and their weights. Note that since the controller is an address, this could in principle implement arbitrary logic, like managing public deposits in a manner similar to a finalized pool. The key difference is that official tooling will not recognize it as a “trustless” pool. Controlled pools with increased trust requirements will be possible with the ❄️Silver Release❄️.
2. *Finalized* pools have fixed pool asset types, weights, and fees. Crucially, this enables `join` and `exit` to be publicly accessible in a safe, trustless manner while keeping a minimal implementation.

Swaps and Exit Fees

The 🍂Bronze Release🍂 charges fees in two situations: When traders exchange tokens (via `swap` and its variants), and when liquidity providers remove their liquidity from the pool (via `exit` and its variants).

Both of these fees are configurable by the controller, but they are also fixed when the pool becomes finalized.

100% of the swap fee goes to the liquidity providers — the amount of the underlying token that can be redeemed by each pool token increases.

Most of the exit fee is returned to the liquidity providers who remain in the pool.
This is similar in spirit to a swap fee charged for exchanging pool tokens with underlying tokens.

The rest of the exit fee is transferred to an account controlled by Balancer Labs, Inc, for the development of ❄️Future Releases☀️.

References

[0] Vitalik Buterin: Let’s run on-chain decentralized exchanges the way we run prediction markets (https://www.reddit.com/r/ethereum/comments/55m04x/lets_run_onchain_decentralized_exchanges_the_way/)

[1] Alan Wu: Building a Decentralized Exchange in Ethereum (https://blog.gnosis.pm/building-a-decentralized-exchange-in-ethereum-eea4e7452d6e)

[2] https://uniswap.io/ (https://uniswap.io)

[3] Zargham, M., Zhang, Z., Preciado, V.: A State-Space Modeling Framework for Engineering Blockchain-Enabled Economic Systems. New England Complex Systems Institute (2018) (https://arxiv.org/pdf/1807.00955.pdf)

Bancor Protocol

Continuous Liquidity and Asynchronous Price Discovery for Tokens through their Smart Contracts; aka “Smart Tokens”

Eyal Hertzog, Guy Benartzi & Galia Benartzi

May 30, 2017

Draft Version 0.99

The phrase "double coincidence of wants" was coined by Jevons (1875). "The first difficulty in barter is to find two persons whose disposable possessions mutually suit each other's wants. There may be many people wanting, and many possessing those things wanted; but to allow of an actual act of barter there must be a double coincidence, which will rarely happen."

Table of Contents

The Bancor Protocol

Abstract: The Bancor protocol enables built-in price discovery[1] and a liquidity mechanism for tokens on smart contract blockchains. These "smart tokens" hold one or more other tokens in reserve, and enable any party to instantly purchase or liquidate the smart token in exchange for one of its reserve tokens, directly through the smart token's contract, at a continuously calculated price, according to a formula which balances buy and sell volumes.

The Bancor protocol is named in honor of the Keynesian proposal[2] to introduce a supranational reserve currency called Bancor to systematize international currency conversion after WWII.

Background

We live in a world where anyone can publish an article, song or video, create a discussion group and even run an online marketplace. We are now beginning to witness the emergence of user-generated currencies. Different types of stored-value ("currencies" hereafter) have been issued and circulated for centuries in the form of bank notes, bonds, equity, gift cards, loyalty points, community currencies[3] and others. Bitcoin was the first ***decentralized*** digital currency, followed by a wave of new cryptocurrencies that have been issued since, and recently we've seen the rise of a new asset class of "tokens" that are typically issued in crowdsales ("ICOs") through smart contracts.

However, currencies, which are essentially networks of value, do not connect to each other in the same way that information networks do. While the switches on Internet exchange points (IXs) interlink information networks, active traders on ***exchanges*** are effectively interlinking currencies.

The current exchange model for currencies/assets has a critical barrier, requiring a certain volume of trading activity to achieve market-liquidity. This inherent barrier makes it nearly impossible for small-scale currencies (such as community currencies, loyalty points or other custom tokens) to be linked (exchangeable) to other popular currencies using a market-determined exchange rate.

In the age of smart contract blockchains, tokens can be automatically managed by immutable code which controls their issuance and behavior. We realized this could mean allowing tokens to hold balances of other tokens (i.e. "reserves"), directly through their smart contracts, that could be designed by their creators and managed programmatically. These new technological capabilities warrant rethinking of the possible solutions for converting one currency to another and determining market prices.

Introducing Smart Tokens: A Solution to the Liquidity Problem

Smart tokens are standard ERC20 tokens which implement the Bancor protocol, providing continuous liquidity while automatically facilitating price-discovery. The smart token's contract

[1] https://en.wikipedia.org/wiki/Price_discovery
[2] https://en.wikipedia.org/wiki/Bancor
[3] https://en.wikipedia.org/wiki/Community_currency

instantly processes ***buy*** and ***sell*** orders, which drive the price-discovery process. Due to this capability, smart tokens do not need to be traded in an exchange in order to become liquid.

A smart token holds a balance of least one other ***reserve token***, which (currently) can be a different smart token, any ERC20 standard token or Ether. Smart tokens are issued when purchased and destroyed when liquidated, therefore it is always possible to purchase a smart token with its reserve token, as well as to liquidate a smart token to its reserve token, at the current price.

A New Method for Price Discovery

A smart token utilizes a novel method for price-discovery which is based on a “Constant Reserve Ratio” (CRR). The CRR is set by the smart token creator, for each reserve token, and used in price calculation, along with the smart token’s current supply and reserve balance, in the following way:

$$Price = \frac{Balance}{Supply \times CRR}$$

This calculation ensures that a constant ratio is kept between the reserve token balance and the smart token’s market cap, which is its supply times its price. Dividing the market cap by the supply produces the price according to which the smart token can be purchased and liquidated through the smart contract. The smart token’s price is denominated in the reserve token and readjusted by the smart contract per each purchase or liquidation, which increases or decreases the reserve balance and the smart token supply (and thus the price) as detailed below.

When smart tokens are purchased (in any of their reserve currencies) the payment for the purchase is added to the reserve balance, and based on the calculated price, ***new smart tokens are issued*** to the buyer. Due to the calculation above, a purchase of a smart token with a less than 100% CRR will cause its price to increase, since both the reserve balance and the supply are increasing, while the latter is multiplied by a fraction.

Similarly, when smart tokens are liquidated, they are ***removed from the supply*** (destroyed), and based on the current price, reserve tokens are transferred to the liquidator. In this case, for a smart token with a CRR less than 100%, any liquidation will trigger a price decrease.

This asynchronous price-discovery model works by constantly readjusting the current price toward an equilibrium between the purchase and liquidation volumes. While in the classic exchange model price is determined by two matched orders in ***real-time***, smart token prices are calculated ***over-time***, following every order.

The above formula calculates the current price, however, when a purchase or liquidation is executed, the effective price is calculated as a function of the transaction size. The calculation can be described as if every transaction is broken up into infinitely small increments, where each increment is changing the smart token’s supply, reserve balance, and thus its price. This ensures that purchasing the same amount of smart tokens in a single or multiple transactions would yield the same total price. Additionally, this method ensures that the CRR will be kept constant and the reserve can never be drained. Essentially, the effect of the transaction size on the price (due to its changing the smart token's supply and reserve balance) is incorporated into the effective price for any transaction. The mathematical functions for calculating price per transaction size are presented further in this document.

Using this method, the Bancor protocol can enable liquidity and asynchronous price discovery for ***existing standard tokens*** -- through smart tokens holding them in reserve, enabling backward compatibility. This use-case and others are described in detail below.

Use-Cases for Smart Tokens

The Long Tail[4] of User-Generated Currencies

The long tail phenomena can be observed in many different online ecosystems such as publishing (blogs), videos (YouTube), discussion forums (Reddit, Facebook Groups) and more. In each of these examples, the long tail has become significantly larger in scale than everything that preceded it. The forming of a long tail begins as soon as the barriers to its existence are removed (e.g. YouTube making it simple for anyone to upload and share user-generated videos).

There are many examples of user-generated currencies, such as group currencies (community oriented currencies), loyalty points (business oriented currencies), and the most recent being hundreds of cryptocurrencies (protocol oriented currencies). However, the need to achieve and maintain liquidity for these small or new currencies remains a significant barrier for their viability.

Smart tokens are unique in that they can be purchased or liquidated by a single party, using the calculated price, ***removing the need for two opposite wants to be simultaneously matched***. This effectively means that by using the Bancor protocol, small-scale currencies with a low expected trade volume can offer continuous liquidity, thus, removing the barrier for them to be linked to the global economy.

Enabling the long tail of currencies is likely to bring about a new generation of creative use-cases. Though it's improbable to predict all of them, some of the more likely use-cases are listed below.

Crowdfunding a Project

The crowdfunding space has been growing rapidly. Smart tokens can be used for crypto crowdfunding initiatives, where the participants receive tokens which are liquid and market-priced. For example, a musician may collect funds to record an album, which would be sold online exclusively in exchange for the issued tokens. A successful album would generate high demand for the tokens, driving up their price and rewarding those holding them. Many other examples exist such as crowdfunding a venture capital fund or raising initial capital for a credit-creating neighborhood currency.

[4] https://en.wikipedia.org/wiki/Long_tail

Token Changers

Token changers are smart tokens that hold multiple reserve tokens, with a total CRR of 100% and can be used to exchange between any standard ERC20 tokens they hold in reserve. A token changer is designed to provide an exchange service between its reserve tokens through a two-step process of purchasing the smart token with one reserve token, and immediately liquidating it for another.

Due to the price calculation formula, each time reserve token X is converted to reserve token Y -- the price of X decreases, while the price of Y increases. Larger transactions will move the price more sharply, however, a higher reserve balance would reduce price volatility.

As noted, any standard ERC20 token can be used as a reserve-token even if it is already traded in other exchanges. In such a scenario, a gap may open between the calculated price of a reserve token and its price in an outside exchange. This situation creates an arbitrage opportunity which ***incentivizes arbitrageurs to restore economic equilibrium***, thus keeping the token changer prices in sync with the prices at which their reserve tokens are traded in other exchanges.

A token changer's creator may set a conversion fee that would apply on each purchase/liquidation. Fees can be accumulated in the reserves and thus increase the smart token's price with every token conversion taking place, increasing the smart token's value. This increase will benefit the holders of the smart token, who may have deposited the original reserves when the smart token was created, or purchased it with any of its reserve token's at any time after that.

Popular exchanges such as MtGox and Bitfinex have been hacked with hundreds of millions of dollars worth of assets stolen from their accounts. Converting one token to another using a token changer does not require depositing funds in an exchange and thus removes the counterparty risk from the process. Another important benefit is that no transaction limits need to be applied, as is the case with other instant trading solutions, due to the decentralized nature of the token changer. While decentralized exchanges offer this benefit as well, smart tokens do not rely on trade volume to provide liquidity.

Decentralized Token Baskets

Smart tokens can be used as decentralized token baskets, which function similarly to ETFs or index funds, simply by holding a portfolio of reserve tokens with a total CRR of 100%. As prices of any of the reserve tokens rise or fall, so does the value of the smart token. Similar to token changers, here as well arbitrageurs are incentivized to realign the conversion rates with market prices which ensures the proper ratios are kept between the reserves according to their real-time market value. These smart tokens enable users to directly hold asset baskets, without a financial services provider as an intermediary.

Network Tokens

A collection of smart tokens that use the same reserve token form a ***network of tokens***. The common reserve token can be described as a ***network token*** which captures the combined value of the network of tokens which hold it in reserve. Increased demand for any of the smart tokens in the network would increase demand for the network token, since it is required for purchasing these tokens, and then held in their reserves. Increased demand drives up the price of the network token, which ***benefits the entire network*** since the value of the tokens' reserves increases, thus to maintain the CRR, the value of the smart tokens also increases. The network token also functions as a "token for tokens", rendering all the smart tokens in the network inter-changeable.

Network tokens can be useful for those who wish to create multiple and related smart tokens for different purposes (e.g. regional network of community currencies, a video game studio with multiple game credits, a group of independant businesses issuing a joint loyalty program). The network token model creates synergetic relationships between the member smart tokens, comparable to the way any single successful Ethereum service can drive up the value of Ether, benefiting ***all of its holders***.

An additional network token use-case is to interlink a set of token changers, each holding a reserve in the network token and a second reserve in another, standard token. This structure would enable exchanging any token in the network to another, while increasing the demand for the network token whenever a new token changer is created or appreciates.

Advantages of Smart Tokens

Smart tokens introduce multiple advantages over the traditional exchange model:

1. **Continuous Liquidity** - Since purchasing and liquidating is done through the smart contract, smart tokens are always liquid, irrespective of their trading volume.
2. **No Extra Fees** - The only mandatory fees applied by a smart token are the blockchain platform fees (gas) which are relatively low.
3. **No Spread** - Since the price calculation is done algorithmically by the smart token, the same price applies for purchasing and liquidating the smart tokens.
4. **Predictable Price Slippage** - Smart tokens allow pre-calculation of the precise price slippage, based on the transaction size, before it is executed.
5. **Lower Volatility -** A smart token with a 10% CRR (for example) is comparable to an exchange with 10% of the ***entire supply*** of a token in its order-book at all times, forming substantial market depth. In a typical crypto-exchange, the share of the supply in the market depth at any given moment is well below 1%. The higher the CRR, the lower the smart token's price volatility. The lower the CRR, the more "new credit" is created relative to the original reserve amount.

The Bancor Protocol Ecosystem

Different parties can take on different roles in the Bancor network ecosystem. The primary forms of participation are as follows:

- **End-Users** can receive, hold, transfer, request, purchase and liquidate smart tokens.
- **Smart Token Creators** can issue new, always liquid smart tokens, that may be used for trading, token changing, as token baskets or as network tokens.
- **Asset Tokenizers** (e.g. Tether-USD, Digix-Gold) can issue ERC20 tokens representing external assets, thus enabling smart tokens to use these assets as reserve tokens. (Existing crypto-exchanges that operate under their local KYC regulations are well positioned to provide asset tokenization services.)
- **Arbitrageurs** are organically incentivized to constantly reduce gaps between prices on crypto-exchanges and the Bancor network. Smart tokens work similarly to exchanges in that purchasing them increases their price and selling them decreases it, so that the same arbitrage mechanics and incentives apply.

A Solution to the Coincidence of Wants Problem

The coincidence of wants problem[5], in the current asset exchange model, creates a situation where assets are required to be traded at a certain minimal volume or else face liquidity risk[6]. The cause for this limitation is that the chance of finding a second party with opposite wants to exchange with, correlates to the asset's trading activity level. Smart tokens solve this problem through the use of reserve tokens which embed market depth directly into the smart token's smart contract.

Smart tokens are a **technological solution** to the ***coincidence of wants problem*** for ***asset exchange***, rather than a labor-based solution as used in traditional (or decentralized) exchanges. The current laborers in asset exchange are the professional market-makers who provide liquidity and facilitate collaborative price discovery. In the domains of information exchange and trade, the technologies of writing and currency replaced labor-intensive solutions (speaking and barter) with technological ones, creating mass efficiencies for societies and unlocking collaboration on a global and intergenerational level. The Bancor protocol proposes to similarly advance the domain of asset exchange by replacing the need for labor with a technological solution to the existing coincidence of wants problem.

Smart Token Initiation and Customization

New smart tokens can be created simply by depositing an initial reserve/s and issuing the initial token supply. Alternatively smart tokens can be initiated through a crowdsale, where a part of the proceeds is allocated as the initial reserve.

[5] https://en.wikipedia.org/wiki/Coincidence_of_wants
[6] https://en.wikipedia.org/wiki/Liquidity_risk

The Bprotocol Foundation

Bprotocol is a Swiss nonprofit foundation whose core objective is the establishment of the Bancor protocol as a global standard for intrinsically tradeable currencies.

By contributing to the Bprotocol Foundation, users will generate BNT - the first smart token to be deployed using the Bancor protocol, establishing the ***BNT network.*** The Foundation will collaborate with different contractors to achieve its goals, as well as governments, businesses, academia and NGOs committed to realizing collaboration potential in communities around the world.

Bancor Network Token (BNT) - The First Smart Token

The BNT will hold a single reserve in Ether. Other smart tokens, by using BNT as (one of) their reserve(s), connect to the BNT network using the price discovery method outlined in this paper. The BNT network will include user-generated smart tokens, token changers (forming a global decentralized, highly liquid exchange), decentralized token baskets as well as subnetworks.

The BNT establishes network dynamics where increased demand for ***any*** of the network's smart tokens increases demand for the common BNT, benefiting ***all*** other smart token***s*** holding it in reserve. Naturally, it is also susceptible to decreased demand. The BNT will be sold in a fundraiser scheduled for June, 12, 2017 10:00 GMT.

BNT Crowdsale Objectives

- A portion of the funds raised will be used as the Ether reserve for BNT (details on the CRR will be outlined in the crowdsale launch announcement), enabling continuous liquidity to Ether for any BNT holder, as well as any holder of a smart token using BNT as a reserve.
- A portion of the funds will be used to develop, promote and support the open-sourced, blockchain-agnostic, Bancor protocol implementations, and support related technologies and applications such as an open-source, user-friendly web service (desktop and mobile) to provide wallet, marketplace, token-conversion, new smart token creation and crowdsale solutions.
- A portion of the funds will be used to set-up and propel the first batch of token changers for popular ERC20 tokens, which function as a ***decentralized solution for token exchange*** between all the included tokens. This model introduces key advantages, incentivizing ***asset tokenizer***s to represent additional real-world assets as Ethereum tokens.
- A portion of the funds will be used to participate in and support innovative and promising future smart token crowdsales in the BNT network. These may include new, location-based and vertical-specific smart token initiatives such as regional token networks, community currencies, crowdfunded projects and other online or offline token-based ecosystems.

Examples and Illustrations

Example #1: Smart Token Transaction Flows

In this example, a crowdsale for a new token (BNT) has collected 300,000 ETH.

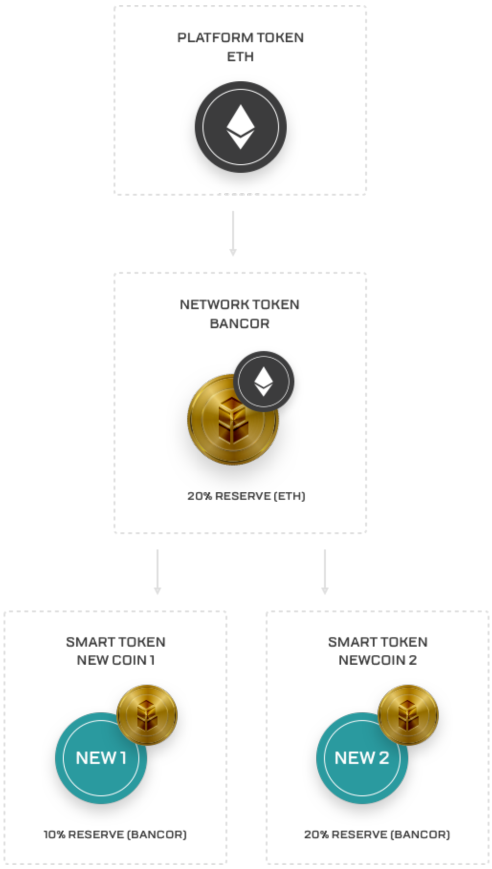
300,000 BNT are issued at a 1:1 ratio and transferred to the crowdsale participants. 240,000 ETH were directed towards funding the BNT project's development and 60,000 (20% CRR) were kept in the BNT smart contract as a reserve.

- Purchasing and liquidating BNT becomes possible as soon as the crowdsale is completed. The opening price is the last crowdsale price, in this example 1 ETH for the first BNT.

- BNT liquidators get ETH from the reserve of BNT, the liquidated BNT are destroyed, and the BNT price decreases respectively.

- BNT buyers get newly minted BNT, their payment in ETH is added to the smart contract reserve and the BNT price increases.

The ETH reserve always remains 20% of the BNT market cap.



Smart Token Symbol	BNT
Reserve Token	ETH (Ξ)
Constant Reserve Ratio (CRR)	20%
Initial Token Price	Ξ1
Crowdsale Proceeds	Ξ300,000
Tokens Issued in the Crowdsale	300,000

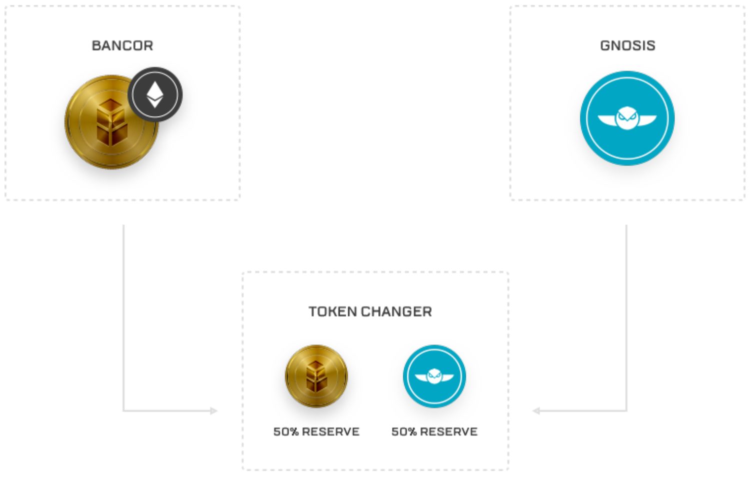
	RESERVE		**PRICING**			**SMART TOKEN**		
Activity	**ETH Recieved (Paid-out)**	**ETH Reserve**	**Effective BNT Price**	**Resulting BNT Price**	**Price Change**	**BNT Issued (Destroyed)**	**BNT Supply**	**BNT Market-cap**
Post-crowdsale initial state		Ξ60,000		Ξ1.0000			300,000	Ξ300,000
300 ETH converted to BNT	Ξ300	Ξ60,300	Ξ1.0020	Ξ1.0040	0.40%	**299**	300,299	Ξ301,500
700 ETH converted to BNT	Ξ700	Ξ61,000	Ξ1.0086	Ξ1.0133	0.93%	**694**	300,993	Ξ305,000
1302 BNT converted to ETH	Ξ(1,308)	Ξ59,692	Ξ1.0046	Ξ0.9959	-1.72%	**(1,302)**	299,691	Ξ298,460
100 ETH converted to BNT	Ξ100	Ξ59,792	Ξ0.9966	Ξ0.9972	0.13%	**100**	299,792	Ξ298,960

Link to Spreadsheet

Example #2: Token Changer Transaction Flows

In this example, a “BNTGNO” smart token is created to function as a token changer between BNT and GNO (Gnosis), holding both in reserve with a 50% CRR each, for a total of a 100% CRR.

Assuming a current market price of 1 BNT = 2 GNO, the contract can define the initial prices as 1 BNT = 2 GNO = 1 BNTGNO and in this example, 10,000 BNTGNO are issued to the depositors of the initial reserves.



- The opening prices are 1 BNTGNO = 1 BNT = 2 GNO as was set in the contract.
- The BNTGNO can be purchased with BNT or GNO.
  The BNTGNO price will increase for the reserve token it was purchased with (BNT or GNO), and decrease in the uninvolved reserve token (due to the increase in the BNTGNO supply).
- BNTGNO can be liquidated back to BNT or GNO, decreasing the BNTGNO price in the liquidated reserve token, and increasing it in the uninvolved reserve token.

This scenario demonstrates how a 100% backed smart token with two 50% CRR reserve tokens can function as a decentralized token changer, open for anyone to use, with its prices organically balanced by arbitrageurs. Both the token changer and the token basket automatically maintain their CRR ratios.

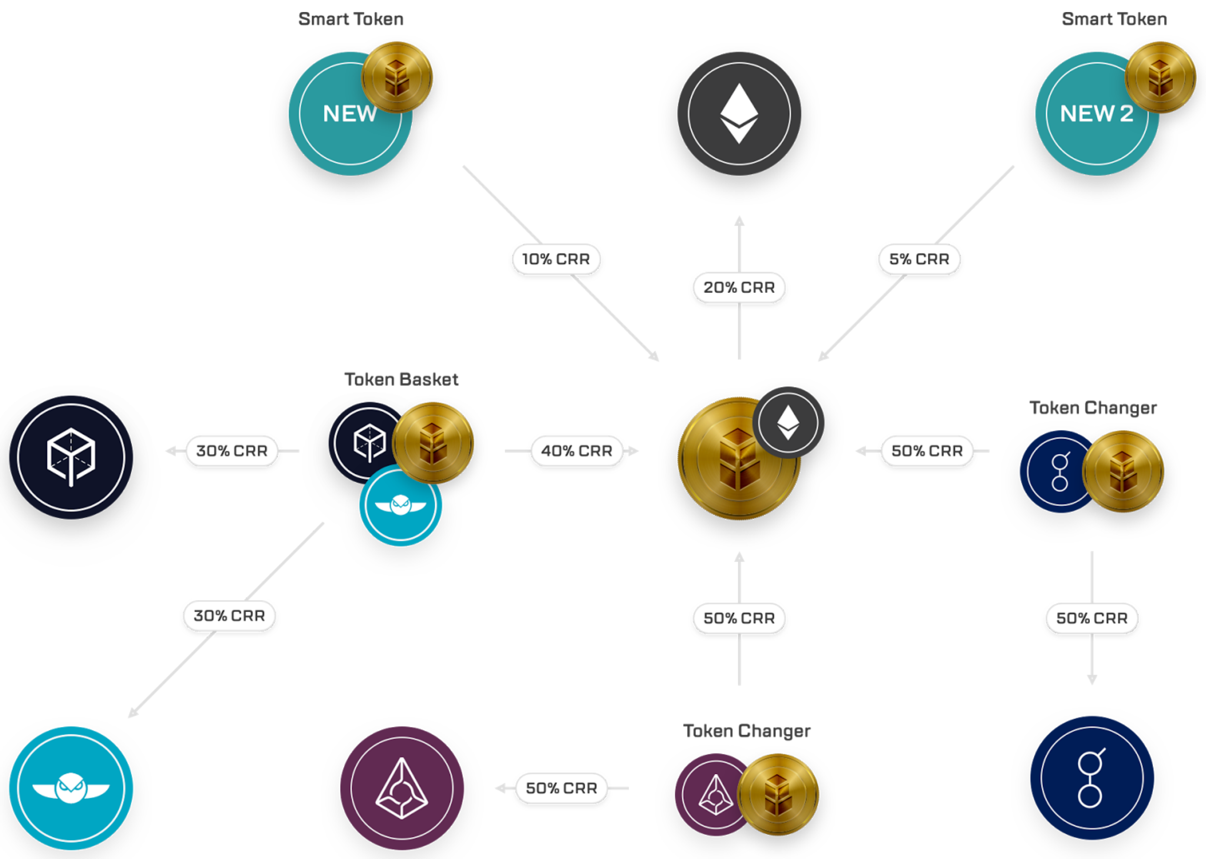
Smart Token Symbol		BNTGNO
Reserve Tokens		BNT + GNO
Constant Reserve Ratio (CRR)	**BNT**	50%
	GNO	50%
Initial Token Price	**BNT**	1
	GNO	2
Deposited Reserves	**BNT**	5,000
	GNO	10,000

		RESERVE		**PRICING**				**SMART TOKEN**		
Activity		**Reserve Recieved (Paid-out)**	**Reserve Balances**	**Effective BNTGNO Price**	**Resulting BNTGNO Price**	**BNTGNO Price Change**	**1 BNT = GNO**	**BNTGNO Issued (Destroyed)**	**BNTGNO Supply**	**BNTGNO Market-cap**
Initial State	**BNT**		5,000		1.000		0.500		10,000	10,000
	GNO		10,000		2.000					20,000
Buying BNTGNO for 30 BNT	**BNT**	30	5,030	1.0015	1.003	0.30%	0.503	**30.0**	10,030	10,060
	GNO		10,000		1.994	-0.30%				20,000
Converting 70 GNO to BNT Step 1 (GNO->BNTGNO	**BNT**		5,030		1.000	-0.35%	0.500		10,065	10,060
	GNO	70	10,070	1.9975	2.001	0.35%		**35.0**		20,140
Converting 70 GNO to BNT Step 2 (BNTGNO->BNT)	**BNT**	(35.0)	4,995	1	0.996	-0.35%	0.496	**(35.1)**	10,030	9,990
	GNO		10,070		2.008	0.35%				20,140

Link to Spreadsheet

Illustrative Map of a Potential Bancor Network

- BNT - The BNT, backed by Ether
- ETH, DGD, DGX, REP and GNT are standard Ethereum-tokens
- NEW - New smart tokens created (e.g. crowdfunding campaign, a community currency, etc.)
- Smart tokens hold reserves (arrows point to the reserve tokens)
- Token changers are 100% backed, and hold two or more reserves



Price Calculation Per Transaction

The actual price of a smart token is calculated as a function of the transaction size.

R - Reserve Token Balance
S - Smart Token Supply
F - Constant Reserve Ratio (CRR)

- T = Smart tokens received in exchange for ***E*** (reserve tokens), given ***R***, ***S*** and ***F***

$$T = S((1 + \frac{E}{R})^F - 1)$$

- E = Reserve tokens received in exchange for T (smart tokens), given ***R, S and F***

$$E = R(1 - \sqrt[F]{1 - \frac{T}{S}})$$

Mathematical proof available[7]

Summary

The Bancor protocol standardizes smart tokens, enabling asynchronous price discovery and continuous liquidity for cryptocurrencies using constant ratios of reserve tokens held through smart contracts, acting as automated market makers. The Bancor protocol enables the creation of hierarchical monetary systems with no liquidity risk. The BNT will be used to establish the first decentralized interconnected currency exchange system which does not rely on matching bid and ask orders, thus remaining liquid irrespective of its trading volume. This system proposes the first technological solution for the ***Coincidence of Wants Problem*** in asset exchange, enabling the long tail of user-generated currencies to emerge.

Acknowledgements

We would like to express our gratitude to the many people who supported us as we wrote this paper. A special thanks to Meni Rosenfeld, Yudi Levi, Amatzia Benartzi, Ron Gross, Assaf Bahat, Sefi Golan, Joshua Alliance, Brian Singerman, Adi Scope, Dory Asher, Tal Keinan, Wings.ai, TheFloor, Arie Ben-David from the Israel Monetary Change Movement, Scott Morris of Ithacash and the Bancor team, Ilana, Asaf, Or, Omry, Itay and Mati. Your support and feedback were truly important to us in improving this document. Thank you.

[7] The mathematical proof is available online at https://goo.gl/HXQBUr

Vesper Documentation

Introduction

Vesper provides a platform for easy-to-use Decentralized Finance (DeFi) products.

Vesper's DeFi products deliver ease-of-use in achieving your crypto-finance objectives. The Vesper token (VSP) is the core economic engine that facilitates the building and expansion of Vesper's capabilities and its community.

The Vesper project rests on three pillars:

Vesper Products: At launch, Vesper offers a variety of interest-yielding "Grow Pools" that enable users to passively increase their crypto holdings by simply selecting the desired aggressiveness of their strategy and the digital asset held. The Vesper Grow Pools represent the first product on the Vesper platform. More will be developed and presented over time.

Vesper Token: VSP incentivizes participation, facilitates governance, and catalyzes user contribution. Users earn VSP through pool participation and, later, participating in Vesper's continuous improvement.

Vesper Community: Vesper is building a user community that sustains and grows the product portfolio, facilitates progressive decentralization, and enables users to build new products while earning a share of that product's fees.

Medium is best for news and insights. Twitter and Telegram serve as our primary notification channels. Discord is where most interactions around governance, community, and development (by the team and community members) will take place.

Vesper Features

Features reflecting the cryptocurrency category's accepted standard and that enable proper interoperability between our platform and others.

- **Non-Custodial:** Assets are deposited to and deployed automatically via smart contracts. Users always maintain 100% ownership of their funds and can retrieve them at any time.
- **Trustless:** Assets are algorithmically deployed through the specifications laid out by Vesper pool strategy smart contracts.
- **Permissionless:** No signup, whitelisting, account verification, or otherwise is required to participate in the Vesper ecosystem.
- **Censorship Resistant:** Users can always interact with the smart contracts directly, which fundamentally cannot be taken down or tampered with.
- **Open Source:** Any developer is welcome to build with Vesper. In fact, it's highly encouraged and heavily incentivized.
- **Fraud Resistant:** The qualities listed above position Vesper's ecosystem to minimize the risk of fraudulent activity typically associated with bordered, custodial, trusted, permissioned, closed source, and censored platforms.
- **Simple, Easy-to-use:** Vesper's user interface was designed to be as seamless as possible. One-click deposit and withdrawals plus mechanisms to reinvest, stake, and harvest.
- **On Ethereum, 'Layer 2 Positive':** The Vesper ecosystem is deployed on the base ('Layer 1') Ethereum blockchain, where it can interact with existing DeFi protocols for yield farming. Layer 2 solutions are under active consideration as potential ways to improve the efficiency of the platform.

DeFi Primitives

Features representing the mechanics of the DeFi products offered as part of Vesper.

- **Grow Pools:** Grow Pools collect a particular asset (ETH, WBTC, USDC, others) via user deposits and deploy the capital to other DeFi platforms as outlined by the Grow pool's active strategies. Yield accrued by these strategies are used to buy back more of the deposit asset, which is delivered to pool participants.

- **Staking Pool (planned):** Token holders can deposit VSP to the *vVSP Staking Pool.* Revenue generated across all Vesper products is used to buyback VSP from the open market. These tokens are delivered to the staking pool, where depositors earn VSP interest proportionate to the size of their deposit.
- **Earn Pools (planned):** Mechanically, Directed Pools operate the same as Grow Pools: deploy deposited assets to defined strategy. However, the yield accrued by Directed Pools is allocated to some other purpose. Some examples include:
  - **Charity Pools:** Yield is delivered to a charitable cause.
  - **VC Pools:** Yield is delivered as venture capital to a startup (likely in exchange for the project's token).
  - **Growth Pools:** Yield from deposit token x is used to purchase token y.
  - **Income Pools:** Similar to Investment Pools, but yield is converted to stablecoin and delivered as a passive income.

All Tokens

Features reflecting all tokens in the Vesper ecosystem, including the VSP governance token and the various tokenized pool shares.

- **ERC20 Standard**: Industry standard for tokens on Ethereum, this enables tokens in the ecosystem to interact with the existing global DeFi ecosystem (Ex: tradeable on Uniswap).
- **EIP-712:** All tokens support EIP-712 for sharing data via message signing. This is an important component of gas-less approvals.
- **EIP-2612 (Gas-less Approvals):** All tokens leverage EIP-2612, which enables gas-less approvals, with the help of EIP-712. Users can send tokens to any contracts after signing an approval message, rather than having to broadcast a transaction.
- **Multi-Transfer**: Inspired by Metronome, all tokens feature a mass pay functionality that enables batched payments in a single transaction.

VSP Token

Features of the VSP token that make it the best token to govern the Vesper ecosystem:

- **Voting Rights:** VSP tokens correspond to the voting weight in the Vesper ecosystem, which includes deployment of reserves and approval of new strategies.

- **Delegation:** Forked from Compound, holders can delegate their VSP voting weight to other accounts.
- **Holistic View:** Vesper is a single-token ecosystem, with every product (new and future) interfacing with VSP. VSP grants voting rights that span the entire Vesper umbrella and revenue generated by all products are used to buy back VSP off the open market.
- **Time-Locked Mintage**: The Administrative "mint" function is locked for the first twelve months. This prohibits a supply expansion beyond 10 million until a point in the future where ownership has fully transitioned to the community of VSP holders, where they can decide for themselves whether or not to extend emissions.

Pool Share Token

Features specific to the various tokenized pool shares that add value + functionality beyond the immediate purpose of tokenized stake.

- **"Lego Brick" Modularity:** Vesper pool shares are designed as a modular asset that can be plugged into other DeFi platforms. Vesper participants maintain liquid ownership of pool shares and can use them for other functionalities while retaining said ownership. For example:
    - **Collateral:** Vesper pool shares can be applied as collateral to create synthetic assets or to be posted as collateral to take out a loan. This is similar to how yCRV is backed by Grow pool shares (yUSDC + yDAI + yTUSD + yUSDT).

Backend Maintenance

Features representing the underlying mechanics that ensure Vesper operates as smoothly and securely as possible.

- **Sweeping:** This is a contract function that swaps non-native ERC20 tokens and deposits them back into the Grow Pool. For example, if the strategy interfaces with Compound, and receives Compound's COMP token, sweeping will liquidate the COMP and reap the profits from it. This also handles any tokens mistakenly sent to the contract.
- **Rebalancing:** Pool assets are redistributed (or rebalanced) on activity. This includes, for example, realizing yield and swapping to deposit asset or adjusting strategy positions on entry to or exit from the pool.

Pool Strategies

Features that guide Vesper Grow Pools to be profitable, secure, and sustainable.

- **Risk Scoring:** Every Vesper Grow Pool has a conservative/aggressive score that reflects the overall risk of the strategies employed by the pool including the security of third-party protocols interacted with, number of contract interactions, and collateralization ratios on loans (if applicable).
- **Modular:** Grow Pool strategies can be modified to integrate additional or alternative actions as well as swapped altogether for better strategies. No action is required on the user's end and funds transition to updated strategies automatically.
- **Upgradeable:** As new and better strategies are proposed within the confines of a pool's defined risk framework, those strategies can be employed without moving funds.
- **Multi-Pool:** Pool assets can be deployed across *n* strategies, with any chose percentage allocated to a strategy (e.g. Allocating 90% of your pool to a Conservative strategy, and 10% to an Aggressive strategy.)
    - **Upgrades:** Upgrades utilize the multi-pool feature to execute a rolling transition from an old strategy to a new one. (Ex: Start with 1%/99% new/old, then 5%/95%, etc. up the staircase until 100%/0%.)
    - **Developer Strategies:** A pool can support an unlimited number of strategies. Therefore, developers may spread funds across *n* pools as a way of testing their strategy.

Web3 UI

Features pertaining to the Vesper frontend to enable a more seamless experience for users.

- **One-Click Reinvest:** Grow Pool users have the option to reinvest their accumulated yield. This means either swapping accumulated VSP for the pool asset or sending VSP straight to the vVSP staking pool.
- **Multi-lingual Support:** Like our pool strategies, website content is modular, and users can interact with Vesper in their native language. As more translations are compiled, they can similarly be added alongside available translations.

Participation Rewards

Features that guide how VSP token rewards are allocated to participants.

- **Merkle Tree Reward:** ZK-Rollups and Merkle trees are employed for distributing VSP to recipients. This enables more sophisticated approaches to VSP distribution (weighted averages, for example) and also eliminates much of the gas burden typically associated with claiming rewards.

Vesper Participants

The following terms outline the participants in the Vesper ecosystem and the roles that they play.

Founders

The team that originally created the Vesper platform. They are compensated with a portion of the originally minted VSP tokens.

Developers

Developers are Vesper community members who contribute strategies to the Vesper platform. They are compensated with a percentage of the fees generated within the strategies they author.

VSP Holders

Members of the Vesper community that hold VSP tokens will be able to cast votes on proposals and receive a share of Vesper revenue by holding and staking VSP tokens.

Pool Participants

Pool participants are Vesper's core users, making them a critical part of the community from Day 1. They often hold VSP tokens, but regardless they have an important voice in the community that is expressed through both their capital allocations and their participation in community conversations.

Multisig Keyholders

At inception, Vesper pool parameters and contract upgrades are managed by multisig keyholders, whose members include the founding team and external partners. Multisig keyholders execute the decisions made by the VSP community.

The initial composition of the Vesper multisig includes founding team members, and will quickly expand to include external partners. You can learn more about Vesper's decentralization roadmap in the section on the Decentralization Plan.

Cybersecurity auditors

Before new strategies are deployed to the Vesper platform, they will need to undergo extensive security audits by professional penetration testing firms. There auditors will be paid with Vesper reserve funds, and will ensure that new contracts are held to the highest levels of scrutiny before users interact with them.

Liquidity Providers

Liquidity Providers assist the Vesper community by providing two-sided liquidity to a VSP pair on the Uniswap platform.

Vesper Grow Pools

About Vesper Grow Pools

IMPORTANT: DO NOT SEND ANY ASSETS DIRECTLY TO HOLDING/GROW POOL CONTRACTS

Vesper Grow Pools (introduced as "Holding Pools") combine diverse deposits into a unified strategy that generates interest to buy back more of the pool’s deposit assets and translate the accumulation into passive returns for pool participants.

Pools are differentiated by deposit asset, strategy type, and risk level. At launch, there are three pools supporting conservative strategies using ETH, wBTC, and USDC. More pools and supported assets will emerge over time, which will be presented to the Vesper community as beta pools prior to their public unveiling on the Vesper website.

The user experience takes what is, as of this writing, a largely manual and time-consuming process and reduces user-facing complexity to:

1. Select strategy.
2. Select Pool.
3. Deposit crypto asset, such as ETH, wBTC, or USDC.

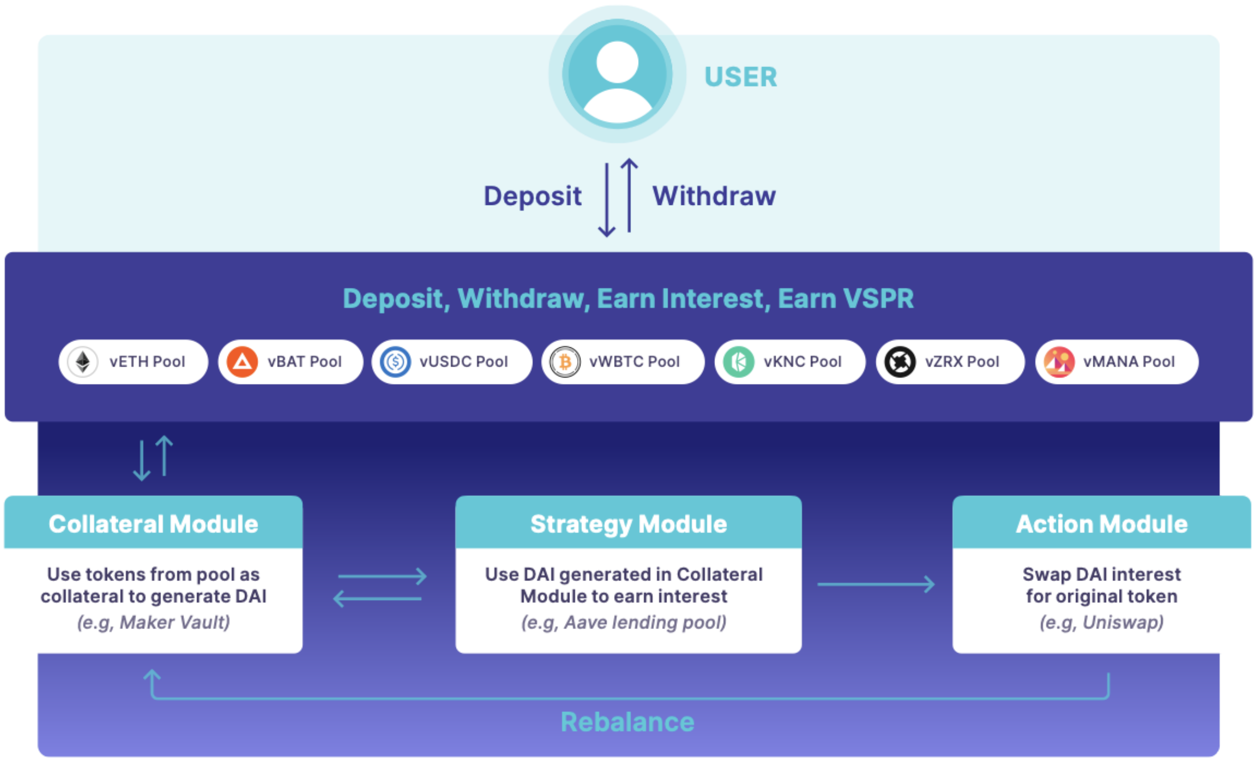
Understanding Vesper Grow Pools

Unlike other yield farming contracts, Vesper Grow Pools emphasize the deployment of deposited assets into third-party DeFi products that generate interest with the goal of growing those deposited assets.

Funds in Vesper pools are used to borrow, lend, and farm yield across various DeFi projects.

Users will select a pool that gives them the asset they want and fits their risk tolerance.

Vesper Pool Mechanics



Vesper Grow Pool Structure

The Grow pool has three main modules: **Collateral**, **Strategy**, and **Action**.

Collateral is where funds are handled when they are deposited and withdrawn. It reflects the contract calls required to move deposits to where they will earn yield. This may be lending platforms like Maker, Aave, or Compound. The process of withdrawing funds through the collateral module can require more effort than depositing them. If the pool is taking out loans with the pooled asset (ETH for example), a partial refund of its outstanding loans may be required before a withdrawal can be executed. This may lead to rebalancing, described below under the heading 'Rebalancing Pools'.

The **Strategy** module deploys capital to generate interest. Depending on the pool, this module could look simple (take out a loan and deposit it somewhere else) or complicated (fractional loans for compounded interest). The strategy might only use the deposit asset deposited in full to an interest-earning DeFi protocol.

Interest accrued is goes into the **Action** module. The action module determines how often to claim interest, and whether to move that interest to the strategy module to be redeployed for further yield, or to take the deposit asset off the table by feeding it back to the collateral module to withdraw it.

Using Vesper Pools

Vesper Grow pools are designed for accessibility. Connect your wallet (e.g. MetaMask) with any of the available deposit assets, and make your deposit through the Vesper web interface. When you deposit, you'll receive a vToken that represents your stake in the pool (e.g., deposit ETH and get vETH). As the pool accrues profit and purchases more of its asset, that tokenized stake will grant more of the underlying asset.

To exit the pool, simply send your vToken back, and you will receive the underlying asset. While your funds are in the pool, you are free to move your tokenized stake to other wallets you control, perhaps to deposit it into another interest-generating strategy.

Fees

The fees work as follows:

- There is **no fee to deposit** into Vesper pools (beyond Ethereum gas fees).
- There is a **0.6% withdrawal fee** when exiting the pool.
- There is a **15% platform fee** collected from accumulated yield.
- For strategies, 5% of that 0.6% withdrawal fee and 15% platform fee is allocated as the Developer's Fee. This is paid to whoever built that strategy. (More in "Developer Incentives.").
- 95% of the accumulated fees are sent to the **Vesper Treasury Box**. The treasury will convert pool shares to buy-back VSP governance tokens off the open market. VSP tokens that are bought back are distributed to the vVSP vault stakers.

Rebalancing Pools

Certain Vesper Pool strategies take advantage of peer-to-peer lending platforms (Maker, Aave, or others) which offer over-collateralized loans. Rebalancing maximizes the loans taken out while remaining within the bounds of safety.

Rebalancing works as follows:

- Each pool has a 'high water' and 'low water' collateralization percentage that correlates to the collateralization requirements enforced by the lending platform.
- There is a `RebalanceFriction` administrative parameter that protects the pools from excessive rebalance calls.
- Any member of the public, user or not, may call the rebalance function every `RebalanceFriction` number of seconds. When the rebalance function is called, if the assets are in 'high water', more loans are taken out. If the pool is at 'low water', some of the loan will be returned to partially close out the loan.
- This rebalance function also claims interest, and swaps interest to collateral tokens.

Collateral Management

In order to maximize profit while avoiding unnecessary liquidation fees, Vesper pools utilize the rebalancing high water and low water variables to guide collateral management.

When users deposit to the pool, their assets are posted as additional collateral as outlined by the strategy, enabling the pool to take out more loans. Likewise, withdrawals remove collateral. Whenever users interact with the pool, or the pool takes profit, it determines the current collateralization ratio and compares it to ratios marked as the high water and low water marks.

If the pool's collateral is at or below the 'low water' ratio, some capital is returned in order to partially close the loan, increasing the collateral relative to the outstanding loan and reducing risk.

Conversely, if it is at or above the 'high water' level, additional loans are taken out.

Rebalance Mechanics

Users are incentivized to call the rebalance function to maintain the health of the pool's outstanding loan portfolio (if applicable to the strategy). The initial three conservative pools use 250% and 275% collateralization benchmarks as the boundaries for low water and high water variables, respectively.

When the collateral on outstanding loans is >275% at time of call, additional loans are taken out to bring the collateral ratio back down. If outstanding loans are <250%

collateralized, then the loans will be partially repaid to bump the ratio up into a healthy midpoint between low water and high water.

Strategies

Each Vesper Grow strategy represents some combination of interactions across various DeFi platforms. This includes, but is not limited to: MakerDAO, Aave, Compound, and Yearn.

Each strategy is differentiated by:

- Deposit asset (that is, the token you can deposit into the pool, such as ETH, wBTC, etc.)
- Contract risk level
- Susceptibility to realized losses

Strategies are classed as medium-risk or high-risk. The risk level reflects the level of safety with regards to the underlying contract interactions. Medium-risk strategies have fewer interactions with safer, audited platforms. (These strategies are considered 'medium' rather than 'low' risk because nothing within the cryptocurrency or DeFi category is truly low-risk.) High-risk alternatives may reflect a higher number or more complex interactions as well as exposure to unaudited contracts, such as Yearn.

Lastly, strategies are classed as Conservative or Aggressive depending on their likelihood of realizing losses. This primarily refers to the susceptibility of outstanding loans to liquidation. Conservative strategies use higher collateral ratios on loans to better protect against 'black swan' events that can jeopardize the loan.

Maker-to-Lending-Platform Strategy

One strategy that may be utilized by the first pools involves MakerDAO plus either Aave or Compound. This strategy operates as follows:

1. The pooled asset is deposited to a Maker vault as collateral
2. DAI loans are taken out against the collateral
3. DAI is deposited to Aave or Compound, where it generates interest
4. The interest is withdrawn and
    1. Swapped on the open market back for the deposit asset
    2. Redeposited to Aave/Compound for compounding returns

This strategy is **medium-risk.** It can be deployed as either **aggressive** or **conservative**, depending on the low-water and high-water collateralization benchmarks. At launch, all three pools are conservative with a low water mark of 250% and high water mark of 275%.

Direct-to-Lending-Platform

Some assets will earn higher APY if deposited straight to lending platforms (Aave and Compound) rather than deposited to Maker for a DAI loan.

This more straightforward strategy comprises of the following:

1. Deposit 100% of the pool asset straight to Aave or Compound (wherever yield is highest)
2. Claim-and-redeposit interest as it is accrued

This is a **medium-risk**, **conservative** strategy; the contracts it interacts with are well audited, and the collateralization ratios are conservative.

It could be used as a launchpad for more aggressive strategies, by using the deposits to Aave/Compound as collateral for loans which are then deployed in other interest-yielding platforms.

Audits & Due Diligence

Security Audits

Vesper Pools have undergone two rounds of independent audits from Coinspect and Certik.

Vesper Pool Contracts

> ⚠ Always interact with Vesper via the Vesper web application. Do not attempt to interact with the Vesper contracts directly.

Vesper pool contract addresses can also be found in our GitHub repository.

Administrative Contracts

Contract	Contract Address
VSP Token	0x1b40183EFB4Dd766f11bDa7A7c3AD8982e998421
Collateral Manager	0x8d0b8e2b5584cE1487317f81Da7d97397eF3e899
Controller	0xa4F1671d3Aee73C05b552d57f2d16d3cfcBd0217
Revenue Splitter	0x097ee00F42f9D7512929A6434185Ae94aC6dafD7

Governance and Revenue Pool

Pool	Contract Address
vVSP	0xbA4cFE5741b357FA371b506e5db0774aBFeCf8Fc

Grow Pools

Pool	Pool Contract Address
vDAI	0xcA0c34A3F35520B9490C1d58b35A19AB64014D80
vETH	0x103cc17C2B1586e5Cd9BaD308690bCd0BBe54D5e
vWBTC	0x4B2e76EbBc9f2923d83F5FBDe695D8733db1a17B
vUSDC	0x0C49066C0808Ee8c673553B7cbd99BCC9ABf113d

Strategies

Pool	Strategy	Strategy Contract Address
vDAI	Compound	0xC80573C8D53Ea1bBa1ED505BBB537DCd4adb9067
vETH	Aave-Maker	0x2010741f855d3CF16FD60e9cce14AF6DE9b526ff
vWBTC	Aave	0x949424E8ef3A9fB1859e4A2fEA8b891bc4D28385
vWBTC	Aave-Maker	0xa3F6Ea08d4083095ec4c363d9cBc629b85029490
vWBTC	Compound	0x040865b75B176278857F459E940b1b8dBF02B62f
vUSDC	Aave	0x3a51F72104fd7c9257730C437B250E99516202Fc
vVSP	vVSP Strategy	0xfd61f9C0796D917466E3aB5f2A40984Fc15794B6

Pool Rewards

Pool	Rewards Contract Address
vETH	0x93567318aaBd27E21c52F766d2844Fc6De9Dc738
vWBTC	0x479A8666Ad530af3054209Db74F3C74eCd295f8D
vUSDC	0xd59996055b5E0d154f2851A030E207E0dF0343B0

GYSR Contracts

Platform	GYSR Contract Address
SushiSwap	0xE07141Bd2De713dF96EA30bFf73eD64fdE560595
Uniswap	0xFc064D2f178f95C86F0901B1700CD99d01968b44

Discussion of Risk

The primary risk faced by Vesper pools is a 'black swan' event, where a pool's underlying asset sees a rapid flash crash. In extreme cases, the debtor will not be able to modify their loan fast enough to avoid liquidation. This is a broader risk that affects DeFi lending protocols as a whole.

In the worst case scenario, a partial liquidation is enforced by the lending protocol. For example, Maker currently carries a 13% fee on the capital liquidated. This would reflect a loss to pool participants.

Vesper pools rebalance their loans algorithmically along parameters specified by each pool. Conservative and Aggressive pools are differentiated by the benchmarks used to take additional loans (when sufficient capital is deposited or the underlying asset appreciates) and partially refund outstanding loans (when capital is removed or the underlying asset depreciates). In this sense, users can further mitigate the risks outlined above by electing to participate in Conservative variants of each pool. (Note that only Conservative pools will be available at the beta launch, with more pools following.)

This risk is further mitigated by the stablecoin offerings. There is no 'volatility' risk with stablecoins apart from the doomsday scenario in which they lose their peg. Such an event would be wholly unrelated to the Vesper ecosystem.

Medium-risk pools only interact with Maker and Aave (and possibly Compound), but high-risk pools may interact with Yearn vaults or directly with other yield farming protocols. These funds are at risk if those protocols are exploited or hacked.

VSP Economics

VSP Token: Supply, Issuance, & Rewards

In addition to the yields generated by the pools, users can also earn the native VSP token. Users earn VSP tokens in three ways: participating in Vesper pools, providing liquidity, or staking VSP. These are described below.

VSP are dripped to users (a little is issued to them every block), and held by the smart contract until the user withdraws from the pool, which triggers the contract to deliver the VSP to their address.

Earning VSP: Vesper Grow Pools

Each Vesper Grow pool is assigned an amount of VSP tokens, and these are distributed to participants proportionate to size of their stake in the pool. Initial pools are incentivized for twelve months after launch.

These VSP tokens are an extra reward on top of the passive yield the pool generates.

Earning VSP: Staking VSP in the vVSP Pool

Users can deposit their tokens to the VSP treasury pool. Just as depositing your ETH in the ETH pool creates vETH, depositing VSP tokens in the VSP treasury creates vVSP.

After withdrawal and yield fees are collected in the pool (ETH, BTC, USDC, etc.), they go to the Treasury Box and are used to buy back VSP on Uniswap. This VSP is delivered to vVSP pool participants as yield on their deposit.

Earning VSP: Liquidity Provision

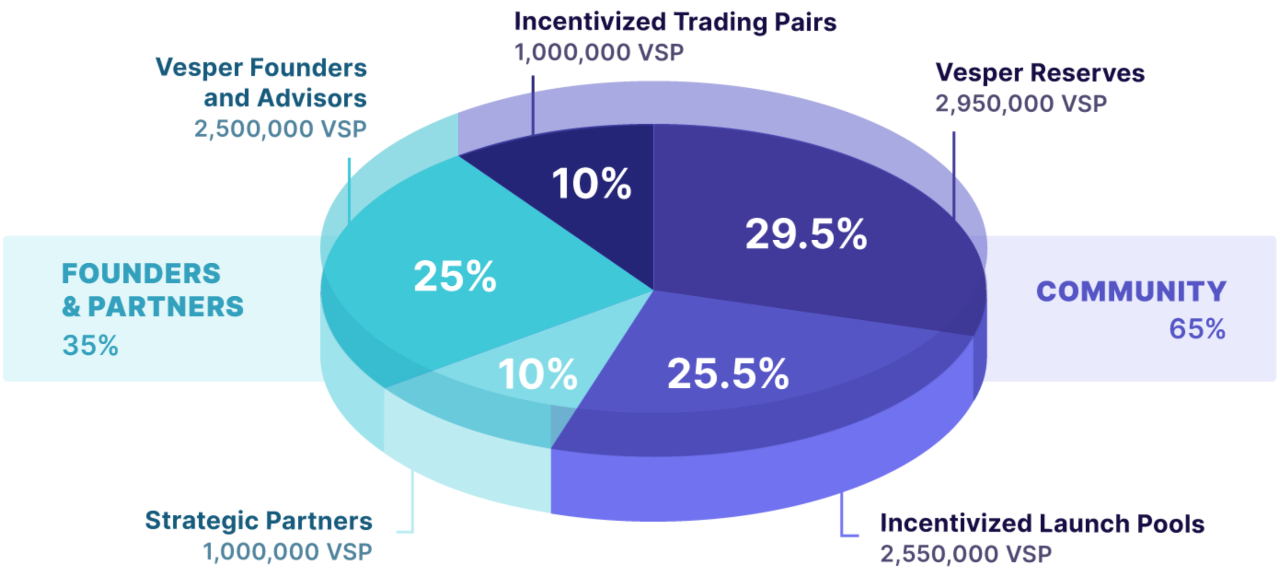
In addition to VSP farming on Vesper pools, liquidity providers to VSP-ETH market pairs can also earn VSP rewards when they stake their tokens into the corresponding Vesper liquidity pool.

The token pairs will be incentivized for the first twelve months. Any extensions beyond the first twelve months will be decided upon by VSP holders.

Supply Dynamics

At launch, Vesper will have a total supply of **10,000,000 VSP**:

- **6.5M VSP (65%) goes to the community**, which includes 2.55M for the Incentivized Launch Pools, 1M to incentivize liquidity providers, and 2.95M to the Vesper Reserves.
- 3.5M VSP (35%) is for Vesper team, advisors, and strategic partners, which is subject to vesting over twelve months.



The initial policy is to keep the supply at 10 million. Twelve months following public launch (after open beta), the community may vote on whether to burn the admin keys and fix the supply at 10 million forever, or, assign minting ability to a DAO for future management. It is up to the community to decide whether limiting future minting benefits the overall Vesper project. For the first 12 months, any amount of VSP beyond 10 million is locked.

Beta Program

Before the official Vesper launch, there will be a brief beta program where users can deposit funds and participate in the initial Vesper pools. There is no active VSP drip during the beta (as there is after), but average balances across the entire period will be recorded and beta users will receive VSP proportionate to their deposited assets after the conclusion of beta.

450,000 VSP is allocated as beta rewards to be distributed to early users of Vesper's open beta. These rewards will be delivered as a multi-send airdrop to each beta participant's wallet on launch day. No further actions are required for beta participants to be eligible.

Token Emissions

Please note that these emissions are *not* finalized, and may be subject to change at the discretion of the community. (For more on Vesper's governance, refer to the documentation's chapter on Vesper's decentralization plan.)

- **Launch Pools** (2,550,000 VSP)
    - 450,000 allocation to pre-launch Beta participants, airdropped at launch
    - 2,100,000 over twelve months, heavily weighted towards the first three months
- **Reserves** (2,950,000 VSP)
    - Reserves for ecosystem growth and developer/community incentives, as determined by VSP voters
- **Liquidity Providers** (1,000,000 VSP)
    - Incentivization for LPs on Uniswap, SushiSwap, 1inch and Loopring (with SushiSwap retaining majority)
    - 1,000,000 distribution over 12 months, heavily weighted towards the first month (and more acutely — the first weeks)
- **Team and Advisors** (2,500,000 VSP)
    - 208,333 (1/12) unlocked at launch
    - 2,291,667 (11/12) vested with streaming unlock (constant drip each block) for eleven months following launch
- **Strategic Partners** (1,000,000 VSP)
    - 83,333 (1/12) unlocked at launch
    - 916,667 (11/12) vested with streaming unlock (constant drip each block) for eleven months following launch

Emissions Breakdown

SushiSwap LP rewards for the first month were distributed via Geyser (gysr.io). Months 2-12 are distributed through a merkle claims process at Pure Finance for liquidity providers who deposit their LP tokens to SushiSwap Onsen (thus receiving $SUSHI rewards as well).

Uniswap LP rewards are distributed via Geyser for the full twelve month duration following the schedule below. You can interact with the Uniswap Geyser here.

1inch LP rewards are distributed through 1inch multi-farming support for those who deposit to 1inch VSP-1inch LP farming. They receive VSP in the same manner as they received 1INCH initially, and 1inch team also has the opportunity to extend their supplied 1inch rewards as well.

Loopring LP rewards are handled through Loopring directly. VSP is rewarded to market makers on the orderbook exchange and follows a formula that multiplies the volume of a user's buy/sell orders by time-spent within 2% of the market price. You can learn more about Loopring's orderbook liquidity mining in the post here.

Additionally, users can earn LRC when they trade across the VSP-ETH AMM. A VSP/DAI orderbook will be included soon as well.

Note that the sum of SushiSwap and Uniswap LP rewards is 1,000,000. **The additional 107,800 VSP across 1inch and Loopring was funded out of the reserves allocation**.

Team, Advisor, and Partner tokens are held in a Sablier contract. Recipients can interface with the contract to claim VSP as frequently or as seldom as they prefer. They receive 1/n of their total VSP allocation over the entire n blocks of their vesting period.

Below outlines VSP emissions. Several assumptions are made:

- All tokens are claimed as soon as they are available.
- No additional VSP is granted from reserves.
- Existing reserve holdings are not counted towards emissions.

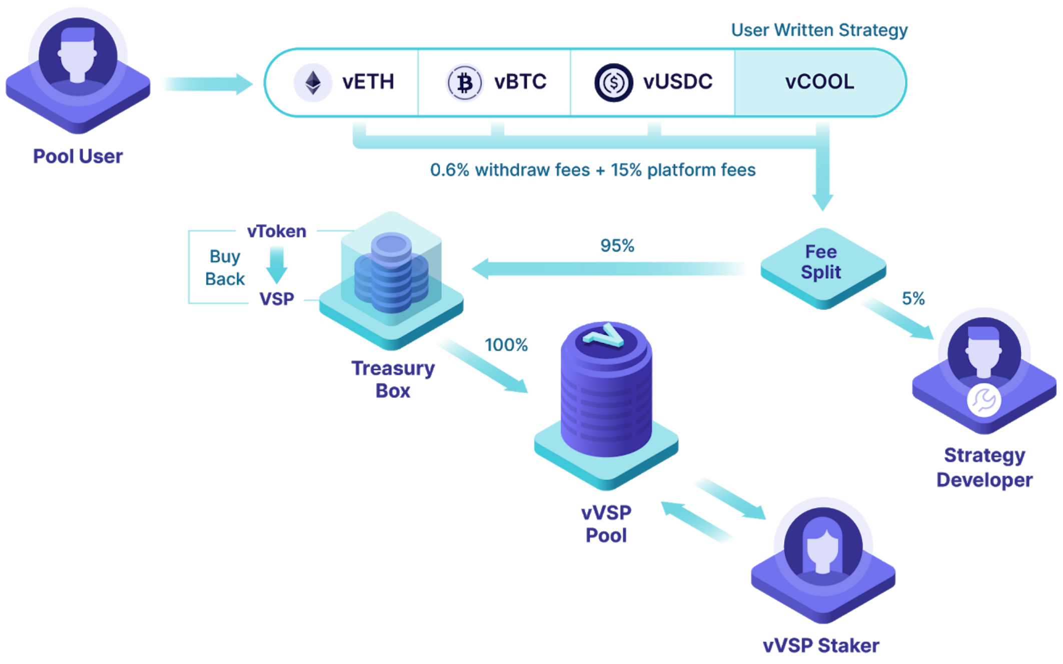
	TOTAL	Month 1	Month 2	Month 3	Month 4	Month 5	Month 6
Team	2000000	333334	166667	166667	166667	166667	166667
Partners & Advisors	1500000	250000	125000	125000	125000	125000	125000
vETH	700000	125,000	150,000	100,000	60,000	45,000	35,000

vUSDC	700000	125,000	150,000	100,000	60,000	45,000	35,000
vWBTC	700000	125,000	150,000	100,000	60,000	45,000	35,000
Sushi	600000	111000	75000	69000	54000	48000	36000
Beta Rewards	450000	450000	0	0	0	0	0
Uniswap	400000	74000	50000	46000	36000	32000	24000
1inch	100000	0	16666	16666	16666	16666	16666
Loopring	7800	1300	2600	2600	1300	0	0
Cumulative	7157800	1594634	2480567	3206500	3786133	4309466	4782799

VSP Liquidity Pools

Platform	Pool
SushiSwap	VSP-ETH
1inch	VSP-1INCH
Uniswap	VSP-ETH
Loopring	VSP/ETH

Revenue Model



Grow Pools

Fees

There is a 0.6% fee on withdrawals, and a 15% platform fee on yield.

For community pools, a 5% share of both of these fees goes to the developer who authored the strategy. This is paid in the pool's asset.

The fees are then directed to the Vesper Treasury Box.

Say a community pool has $50 million in total withdrawals and $5 million in accumulated yield. This pool would pay a 0.6% withdrawal fee ($300,000), and a 15% platform fee ($750,000), for a total of $1,050,000. Of this $1,050,000, the Developer would get five per cent – $52,500 – and the remaining $997,500 would go to the treasury. That $997,500 is converted to VSP via open market buy-backs (see the algorithm explained below), and 100% of that VSP is distributed to vVSP pool participants.

Incentives

Any user may trigger a **Rebalance-Collateral** operation, to address pool-wide systemic risk and generate additional stablecoin yield. If collateral price falls below Low Water, this operation will prevent liquidation. If collateral price rises above High Water, this operation will generate additional DAI, which, in turn, generates more yield for the entire pool. Users have the incentive to use this to earn maximum yield and avoid liquidation.

Any user may trigger a **Rebalance-Earned** operation. This operation swaps earned stablecoin interest for underlying collateral (e.g. DAI to ETH), and adds this collateral to the pool's holdings. Users have the incentive to *not* call the operation, as the longer the collateral stays in there, the more yield it earns. Balancing that, they have the incentive to call the operation prior to withdrawal, to maximize the amount of collateral in pool, to maximize their share of collateral withdrawn.

Treasury Income

Summary

The Vesper Treasury earns the 0.6% withdrawal fee and 15% platform fee described above. Additional income occurs periodically via partnership incentives such as liquidity mining.

All Vesper pools send pool shares to the vVSP pool. These pool shares are distributed according to an algorithm described below. This algorithm will initially be set by the founders, and later transitioned to community control.

Current Algorithm

Once a day or more, the vVSP **rebalance** operation liquidates the next-in-line pool shares to VSP tokens. One pool's shares are liquidated at a time in a round-robin fashion. The frequency of conversion is determined by how quickly the treasury grows (more shares = more frequent conversion). Example: 12.5 vETH tokens unwrap to 15.25 ETH, swapped to 18.9 VSP via Uniswap. Keep in mind that the treasury funds represent the 95% remaining after 5% to the pool's Developer.

That 18.9 VSP is deposited into the vVSP pool, increasing the total value of all vVSP tokens by increasing the NAV of the vVSP pool.

Community Growth Engine

The Vesper project and community are tied together by the VSP token.

At launch, 10,000,000 VSP tokens will be minted, and allocated almost entirely to community-focused efforts. In the short term, 3,400,000 of these funds will be allocated to product and liquidity provisioning incentives.

In the long term, the community reserves (3,600,000 tokens) combines with community-defined level of VSP revenue from the VSP vault. These VSP tokens will be held in a community vault, to be used to create a continuous cycle of bootstrapping new products with VSP incentives, which in turn benefits the VSP token, which generates funds to bootstrap yet more products.

Community Participation & Governance

Decentralization Plan

> *At launch, Vesper network governance relies on a strong social contract between the founding team and stakeholders. Over time, the Vesper platform will mature to become entirely community-driven. This means that Vesper stakeholders will eventually be in total control of the future of the product ecosystem. From Day 1, the long-term objective is to develop a robust decentralized community of which the founding Vesper team only accounts for a small minority of the decision-making authority.*
>
> *Across all phases of decentralization, we have a firm commitment to engaging with VSP holders and community members on all major changes and updates.*

Overview

VSP is the platform's governance token, which gives token-holders the ability to participate in on-chain votes on new pool strategies and other platform decisions. This section addresses how members of the Vesper community might use their tokens in this manner.

Governance Phases & Rationale

Vesper Governance is derived from the Compound governance module, a tried-and-true framework that has become familiar to the DeFi community. From Day 1, the founding team will begin relinquishing control over Vesper operations through phases of progressive decentralization, and will strive for the highest standards of communication and transparency.

Transitioning to Community Governance

Many DeFi projects have experienced early 'growing pains' when initiating community governance. We hope to mitigate this friction with an iterative transition to community governance.

1. For the first 2-6 months, ownership functionalities will be retained by the team’s multisig in order to upgrade strategies, introduce new pools, allocate VSP rewards, and so on.
2. After 2-6 months, governance responsibilities will be transferred in full to holders of vVSP in the vVSP vault.
3. At the end of that 2-6 month period, the Founding Team will stake 1,000,000 VSP from the community reserve to the vault. This will mint vVSP 'receipt tokens' as a 'governance bootstrap' to ensure quorum and good governance principles are met. The voting power of those tokens will be initially delegated to the Founding Team's multisig. As the 1,000,000 VSP that created these voting tokens are unstaked, they will be deposited back into the vault for the benefit of the other vVSP holders.
4. Over the course of one year after the governance module is launched, vVSP delegated to the Founding Team will be forfeited quarterly, so that 100% of the control will be in the hands of the community at the end of this one-year period.

We expect the community’s voting power to exceed that of the team by Month Six.

The revenue generated by the reserve fund is 'community property' and is sent back to the other vVSP holders. The flow of assets will be very visible/auditable to the community.

Votes can only be cast by vVSP token-holders. The vote passage/approval requirements are as follows:

Action	Threshold
Bring a proposal to vote	Submitters must have the delegation of at least 1% of the outstanding vVSP supply.
Voting - Reach Quorum	4% of the outstanding vVSP supply must vote ‘Yes’ on the proposal.
Voting - Vote Passes	A minimum of 50%+1 of votes cast with a minimum of 4% of vVSP supply as ‘YES’ votes .

Voting & Participation

Participants can engage with the Vesper community by proposing, developing and assisting **Vesper Improvement Proposals** [VIPs], casting votes on VIPs, and sharing their opinion in our community chats. (See "The Voting Process.")

The Voting Process

All vVSP tokens are eligible to participate in Vesper governance votes. VSP stakeholders can acquire vVSP by staking their tokens in the vVSP Vault.

Vesper Improvement Proposals (VIPs) are submitted by community members. One function of VIPs will be for adding new, community-built strategies to the platform.

Every Vesper Improvement Proposal must be accompanied by executable code and framed in a Yes/No format. It may include no more than 10 contract actions.

In order for a Vesper Improvement Proposal (VIP) to be brought to an on-chain vote, the proposer must either hold 1% of the vVSP in existence, or get others to delegate their vVSP to reach the 1% necessary.

When a proposal vote is formally initiated, the voting period will begin. The voting period will last for 17,280 blocks, which is roughly 3 days.

In order for a Vesper Improvement Proposal to succeed, it must meet thresholds for quorum and passage:

1. For a proposal to reach quorum, 4% of the outstanding vVSP supply must vote 'Yes' on the proposal.
2. For a proposal to pass, after reaching quorum, at least 50%+1 of votes cast must vote ‘Yes’ on the proposal, and the 'Yes' votes must total a minimum of 4% of the vVSP supply.

Below are examples of different governance scenarios to further illustrate the process:

- If 5% of the outstanding vVSP supply votes 'Yes', and 6% of the outstanding supply votes 'No' (meaning 11% of the outstanding supply participated), the vote would fail because, while quorum was met, there were more 'No' votes than 'Yes' votes.
- If 4% of the outstanding vVSP supply votes 'Yes', and 3.5% of the outstanding supply votes 'No' (meaning 7.5% of the outstanding supply participated), the vote would pass because quorum was met, and there were more 'Yes' votes than 'No' votes.
- If 3% of the outstanding vVSP supply votes 'Yes', and 2% of the outstanding supply votes 'No' (meaning 5% of the outstanding supply participated), the vote would fail

because, while there were more 'Yes' votes than 'No' votes, the quorum requirement of 4% 'Yes' votes was not met.

vVSP tokens can be delegated to another account, lending their voting power to that account.

Governance Principles

The founding team will always provide the community answers to these basic questions for every material decision the team makes:

- What decision has been made?
- Why was that decision made?
- Who is impacted by the decision?
- When is the decision going into effect?
- How can the community provide feedback and voice their opinion?

Founding Principles

As with other projects in the DeFi space, Vesper's governance will emerge from community collaboration and participation. **We begin the project with a few simple governance principles:**

- Any and all material changes to Vesper products and VSP should be proposed in public, with code, with appropriate time for community feedback.
- We believe that a responsible yield farming network can remain in the founding team's hands for the initial phase only, until power can be transferred to the community. Our roadmap for progressive decentralization outlines our strategy for transitioning to fully autonomous and decentralized decision-making.
- We believe a successful governance system should minimize the time and cost necessary for a person or entity to participate in governance. We will explore gas-efficient methods of voting like Layer 2 scaling solutions to reduce costs associated with voting.

Vesper Developers

Developer Incentives

Any member of the community can create and propose new Earn pool strategies and earn revenue for life in doing so. We refer to these people as Developers.

The process works as follows:

1. The Developer creates their pool strategy
2. They propose the strategy to the Vesper DAO
3. The strategy is approved and implemented by the DAO
4. The Developer who authored the strategy earns a cut of the revenue (forever)

When the pool goes live, 5% of all revenue it generates goes to the Developer as a reward. Revenue is made up of the 0.6% withdrawal fee and 15% platform fee on yield accrued on assets deployed by the strategy. More detail under Revenue Model.

Vesper Improvement Proposal Template

Vesper Improvement Proposals (VIPs) can be submitted at [GitHub Link].

VIP-000: Title

A VIP number, like `VIP-001`, will be assigned and the proposal author should give it a short, descriptive title.

Summary

In easy-to-understand language, describe the purpose of your proposal and what it intends to achieve for the Vesper network.

Abstract

Briefly describe what the proposed change will do.

Expectations

Detail the expectations and assumptions behind the VIP's proposed contract. This is the qualitative and quantitative rationale behind the contract's strategy.

Specification

In detailed, technical language, describe the inner workings of your proposed contract.

Test Cases

Describe how other implementations or back-tests of this contract performed.

FAQ

What is Vesper?

Vesper is a platform of DeFi products designed for ease-of-use, longevity, and scale. It is a comprehensive ecosystem governed by the VSP token.

Vesper Grow

What are Vesper Grow Pools?

Vesper Grow Pools are algorithmic DeFi lending strategies. They pool capital from a group of users and deploy it to generate interest across various DeFi protocols. Accrued interest is used to buy back the pool's deposit asset (which may be ETH, BTC, USDC, or something else), and award it as interest to participants.

How do I interact with Grow Pools?

All you have to do is choose the pool you are interested in and deposit your asset. One transaction, and the pool does the rest. Similarly, you can withdraw your funds and claim your interest with one transaction.

What Grow Pools are available today?

As of the launch there are three pools: ETH, BTC, and USDC. We will frequently offer trial pools to the community before they go prime-time on the Vesper website.

Has the code been audited?

Yes. The code has undergone two independent audits by Coinspect and Certik. See the Audits and Due Diligence section for more details.

What is the benefit to depositing my funds in an Grow Pool?

Grow Pools deploy your assets into DeFi lending strategies. You can choose a pool based on your risk tolerance and desired token. This reduces a process that typically comprises more than a dozen fee-extracting transactions, hours of research, and constant monitoring down to a one-time deposit and withdrawal.

Additionally, Grow pool participants farm VSP tokens, further rewarding users with even greater passive returns, catalyzing participation, and forming the basis for progressively decentralized governance.

What happens to my funds after I deposit them in a Grow pool?

Deposits are pooled and deployed through the strategy outlined by the particular Grow pool. See the Strategies page for more information on what that strategy may look like.

Is there any risk associated with Vesper Grow pools?

Grow pools that interface with loans are at risk in the event of a so-called black swan event, such as when a pool asset sees a flash crash in a short amount of time. In this event, the pool's outstanding debt position may become under-collateralized, leaving the lender insolvent, and the pooled funds may be hit with a liquidation fee, which could translate into a loss to the participant.

Each token supported offers a pool that pursues conservative and aggressive strategies. Conservative pools use higher collateralization ratios and are therefore less vulnerable to such a risk – but not completely immune.

Additionally, Grow pool strategies are only as safe as the platforms they interact with. Medium-risk pools only interact with blue-chip DeFi protocols like Maker, Aave, and Compound, but high-risk pools may interface with newer and less established alternatives.

What are the fees?

There is a 0.6% fee on withdrawal from Vesper Grow pools, and a 15% platform fee on yield generated by the deposited assets.

Where do the fees go?

The platform fee and withdrawal fee are both taken in the form of pool shares, and delivered to the Treasury Box. They continue to earn yield as any pool share would, until they are converted to VSP tokens by selling them on the open market via an AMM like Uniswap or SushiSwap. These VSP go to the vVSP pool to be distributed to vVSP holders.

VSP Token

What is the VSP token?

VSP is the governance token that serves as the basis for the Vesper ecosystem. VSP holders can vote on proposals, and additionally deposit their tokens to passively accumulate more VSP.

How do I earn VSP tokens?

There are three ways to earn VSP tokens:

- **Participating in Vesper pools.** Each pool is assigned an amount of VSP tokens that are distributed to participants proportionate to size of stake. Initial Grow pools are incentivized for three months after launch.
- **Providing liquidity.** Liquidity Providers to the VSP-ETH Uniswap pair are incentivized with VSP similarly to the Grow pools. The trading pair is incentivized for one year after launch.
- **Staking your VSP.** Users can deposit their tokens to the VSP treasury pool. A small percentage of withdrawals are allocated to the treasury box, and those funds are used to buy back VSP and award to pool depositors.

How will I receive my VSP tokens?

VSP tokens are 'dripped' to pool participants, and held by the smart contract until they broadcast a transaction to exit the pool, whereupon the VSP is delivered to their address.

What is the vVSP Pool?

The vVSP pool is a revenue-sharing mechanism. It rewards VSP holders with additional VSP when they deposit their tokens in it.

Just like the other pools, where you deposit ETH to get vETH, or USDC to get vUSDC, you can stake VSP and you will get vVSP, a tokenized share of the vVSP pool.

After withdrawal and yield fees are collected in the pool (ETH, BTC, USDC, etc.), they go to the Treasury Box and are used to buy back VSP on Uniswap. Of this VSP, 5% goes to the project's founders, and the rest is delivered to vVSP pool participants as yield on their deposit.

Governance

Who makes decisions about what happens with Vesper?

Initially, the founding team will govern the project. This is a hard decision to make, but seeing how other tokens have had widespread difficulties with community governance on Day One, the safest route seems to be keeping Vesper under our wing immediately after launch. As the Vesper ecosystem matures, governance will soon be turned completely over to the community. See the Decentralization Plan and Roadmap page to learn more.

Who governs the Treasury Box?

Just like other Vesper pools, the Treasury Box has an algorithmically-enforced strategy. At launch, the multisig signers from the Vesper team have jurisdiction to make changes to the treasury strategy. As Vesper transitions to community governance, changes to the strategy are proposed and deployed in the same manner as any of the Vesper pools.

Can I propose new products or strategies to be added to Vesper?

New products and strategies can be proposed by any holder with 1% of the issued vVSP supply. You can hold these VSP tokens yourself, have other VSP holders delegate tokens to table your proposal, or a combination of both.

On day 1, before any meaningful portion of the supply has been issued, proposals will be initiated by the Vesper team. We are eager to transition to community governance, at which point VSP community members will be able to create a formal proposal for a new strategy, product, or change to the ecosystem. The sky's the limit to how many assets and strategies can be deployed, and we are excited to see what the community comes up with.

How do I vote?

All vVSP tokens are eligible to vote. Users can acquire vVSP tokens by staking their VSP in the vVSP Vault. All Vesper Improvement Proposals (VIPs) that are backed by 1% of the total vVSP supply will be voted upon by the the community of vVSP holders. You can learn more about the voting process in the Voting and Participation section.

Glossary of Terms

Medium-Risk/High-Risk – Refers to the security of the Vesper Grow Pool strategies. Medium-risk strategies have less steps and fewer asset conversions. Additionally, medium-risk strategies only interact with audited and highly secure third party contracts. Alternatively, high-risk strategies are more complex in terms of the underlying contract calls and may interact with unaudited protocols, such as Yearn vaults. (There is no 'low-risk' designation because we believe that labeling *anything* as low-risk in crypto is disingenuous.)

Conservative/Aggressive – Refers to how prone a Vesper Pool strategy is to realizing losses due to partial liquidation. Conservative pools adhere to higher collateralization ratios than aggressive pools, and as such are better protected in the event that the pool's deposit asset sees a rapid loss in value (thus putting the outstanding loan underwater and in danger of liquidation).

Low Water/High Water – Vesper Pools that deposit assets to MakerDAO in order to withdraw DAI loans adhere to low-and-high water collateralization targets. If the ratio of collateral to the outstanding DAI loan falls below "low water", the loan will be partially refunded to increase the ratio. If the ratio is above the high water benchmark, additional DAI will be taken out to shift the ratio below "high water".

Black Swan Event – Extenuating circumstance where a drastic "flash crash" in the price of an asset causes financial products interfacing with the asset to breakdown. In the world of cryptocurrency, this looks like a substantial crash in Ethereum, BTC, or other collateral asset that leads to mass liquidations. The danger is compounded with low scalability making it difficult or impossible for debtors to service their loans in order to avoid liquidation. Vesper's additional **low/high water** mechanism further insulates Grow Pools from the detriments of a potential Black Swan.

Rebalance-Collateral – Any user may trigger this operation to address pool-wide systemic risk and generate additional stablecoin yield. If collateral price falls below **Low Water**, this operation will prevent liquidation. If collateral price rises above **High Water**, this operation will generate additional DAI, which, in turn, generates more yield for the entire pool. Users have the incentive to use this to earn maximum yield and avoid liquidation.

Rebalance-Earned – Any user may trigger this operation to swap earned stablecoin interest for underlying collateral (e.g. DAI to ETH), and add the purchased collateral (e.g. ETH) to the total pool holdings. Users have the incentive to *not* call the operation, to maximize stablecoin deployed earning yield. Users have the incentive to call this operation prior to withdrawal, to maximize amount of collateral in pool, to maximize share of collateral withdrawn.

Rebalance – Every six hours, the vVSP operation chooses the largest-valued pool shares in its inventory, and liquidates that to VSP tokens. *Example: 12.5 vETH tokens unwrap to 15.25 ETH, swapped to 18.9 VSP via Uniswap. The VSP tokens acquired during the rebalance operation are then split.*

Developer's Fee – The 5% share of the fees taken by pools (as withdrawal fees and platform fees) that is allocated to the author of the strategy.

vVSP Pool – VSP holders deposit their tokens to the vVSP Pool in order to "stake" VSP. Revenue generated by Vesper products is used to buy-back VSP from the open market, where it is delivered to stakers as deposits to the vVSP Pool, where it is distributed to stakers in proportion to their tokens staked versus total pool size.

Treasury Box – Fees taken from Vesper products (pools or otherwise) are taken as tokenized shares. The treasury box converts these tokenized shares back to the underlying assets then swaps the assets for VSP on Uniswap, where 95% of it is given to stakers and 5% to strategy developers.

Reserves – 2,950,000 VSP of the 10,000,000 total supply is allocated to Vesper's DAO holdings. These reserves can be deployed only with the democratic approval of VSP holders. Reserves are designed to extend incentives to holding and liquidity pools beyond initial allocations and introduce incentives to new products as they are released.

Earning Rate – Earning Rate reflects two figures: the underlying yield accrued by pool assets as they are routed to other DeFi platforms per the pool strategy and the VSP "boost" assigned as part of VSP token distribution. The "**spot**" earning rate is calculated as last 24-hours performance annualized and compounded, while "**average**" reflects the past 30-days annualived and compounded.

vVSP Flow – The sum of VSP bought back over the past 24 hours and delivered to vVSP pool depositors.

renproject / **ren**

Code Issues 6 Pull requests Actions Projects **Wiki** Security

Home

Jump to bottom

Vincent Ward edited this page on Oct 21, 2020 · 17 revisions

RenVM is currently at the beginning of phase sub-zero.

In this Wiki, we present RenVM, a Byzantine fault tolerant (BFT) network that enables universal interoperability between blockchains. By combining consensus with secure multi-party computation (MPC) algorithms, RenVM is able to instantiate a decentralised, permission-less, and trust-less custodian capable of locking assets on one chain and minting one-to-one pegged representations of them on other chains. In this way, users are able to interact with multiple applications, multiple assets, and multiple chains with only one transaction. Throughout this wiki, we will explore how RenVM is able to achieve this, using BTC-on-Ethereum as a particularly interesting case study.

TL;DR

RenVM is a decentralised crypto asset custodian that:

- **enables universal interoperability between blockchains:** anyone can use RenVM to send any asset to any application on any chain in any quantity.
- **has robust security:** large bonds, large shard sizes, and continuous shuffling make RenVM extremely difficult to attack, even for irrational adversaries. In the unlikely event of a successful attack, RenVM can restore lost funds.
- **is scalable:** as more assets are locked into the custody of RenVM, the algorithmic adjustment of fees allows RenVM to automatically scale its capacity to meet demand.
- **provides an optimal user experience:** users can interact with multiple assets, applications, and chains with only one transaction.

Introduction

Blockchains have enabled a new approach to technology and finance, one where users are self sovereign and do not need to trust centralised third-parties or intermediaries. Since their inception, blockchains have found the most adoption in financial applications, allowing users to store and transfer value, purchase goods, and earn interest. In somewhat of a contradiction, most of this activity has taken place on centralised exchanges and websites, where central authorities are able to subvert and censor users. However, in the past few years, the rapidly growing DeFi movement has aimed to empower users by giving them access to all of the same functionality but without the need for centralised third-parties and intermediaries. DeFi encompasses many kinds of decentralised applications, but ultimately it is an attempt to enable sending, lending, exchanging, and leveraging (and more) without the need to leave the Byzantine resistant world of the blockchain.

Now, the struggle has changed. To scale this new class of financial technology (and not in the transactions-per-second kind of way), a major shortcoming of blockchains must be addressed: interoperability. At the time of writing, Bitcoin is 9x larger than Ether, and eclipses all other cryptocurrencies. It is worth more than the next 100 largest cryptocurrencies combined. But, despite its dominance, no general purpose interoperability solution exists that does not require centralised third-parties. Furthermore, this is a deficiency that extends beyond Bitcoin. None of the top ten blockchains (ranked by market capitalisation of their assets) are interoperable with one another.

Interoperability presents a major challenge, but it is critical for the continued growth of blockchains and the financial technologies built upon them. For DEXs to grow, access to more liquid assets is needed. For lending platforms to grow, access to more interesting and diverse assets is needed. For synthetics and derivatives to grow, access to higher market cap assets is needed. For the ecosystem to take the next step, we need to connect our users. Network effects that are achieved on one chain should not need to be replicated on others, applications that are built, battle-tested, and adopted on one chain should not need to be built again, and competition should encourage innovation and improvement instead of cloning. Interoperability will not solve all of the challenges faced by blockchains, but it does solve some of them and lay the foundations for solving many more.

To this end, we have developed RenVM — the design of which will be the focus of this wiki — with the intent to bring interoperability to all blockchains, developers, and users. RenVM is designed with careful consideration for simplicity of use, and is able to offer a *native* user experience, where users are only ever required to make a single transaction from a single chain. Such a transaction can kick-off arbitrarily complex logic that spans many applications, many assets, and most importantly, many chains. We call this universal interoperability.

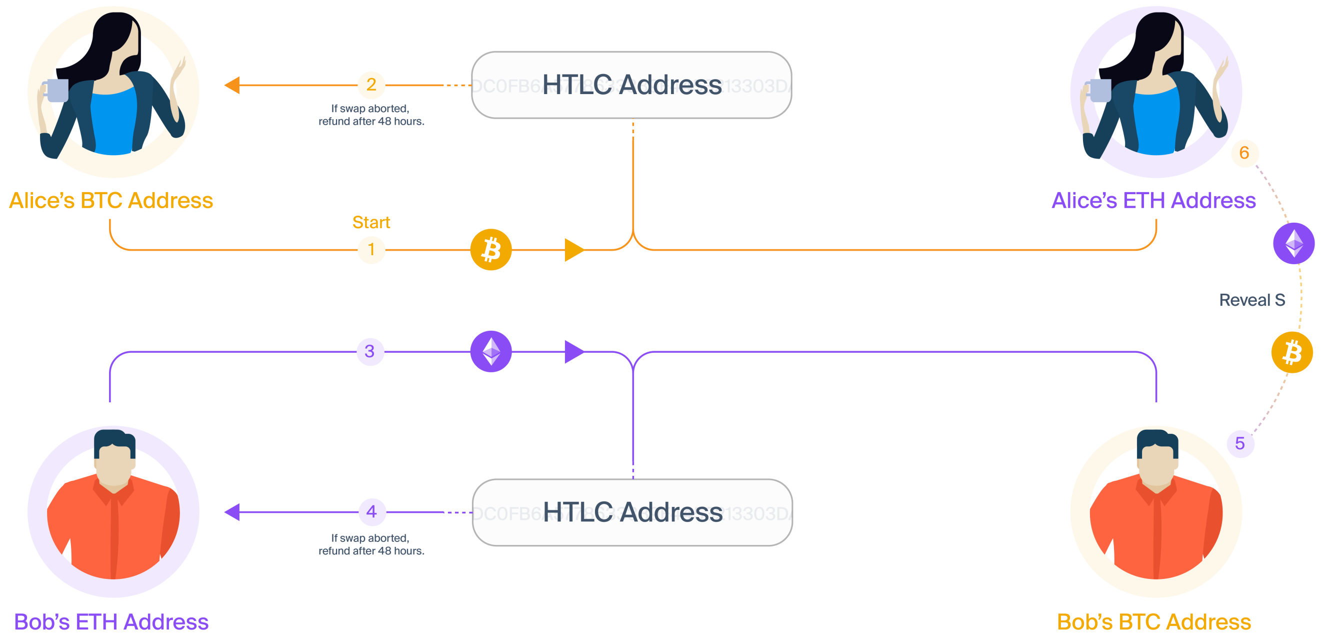
Universal Interoperability

Interoperability is quite an overloaded term, used to describe many different kinds of functionality. We will dive into some of these definitions in a moment, but first, it is worth giving our own definition to provide clarity on exactly what RenVM is built to accomplish.

We define *universal interoperability* as the ability to send any asset from any chain to any other chain for use in any application. Furthermore, we require that such a universal interaction, spanning multiple assets, chains, and applications, must be executed as the result of one transaction made by the user. For example, a universal interoperability protocol must allow a user to 1) exchange BTC for ZEC on an Ethereum DEX, 2) send that ZEC to Polkadot where it is used to collateralise a stable-coin, and 3) send that stable-coin back to Ethereum where it is lent out to another user. All of this must happen as a result of only one Bitcoin transaction made in the first step. It is worth explicitly noting that universal interoperability protocols must not make any assumptions about the specific applications that will be using them. Many of the decentralised applications available today could not have been imagined when blockchains were first brought to the world by Satoshi Nakamoto, and a universal interoperability protocol must ensure that it works for use cases that we still have not imagined today.

Related Work

There have been many attempts to achieve various forms of interoperability between blockchains, most of which have focused on interactions between Bitcoin and Ethereum. In this section, we will discuss some of the existing solutions that have been proposed and look at their main disadvantages.

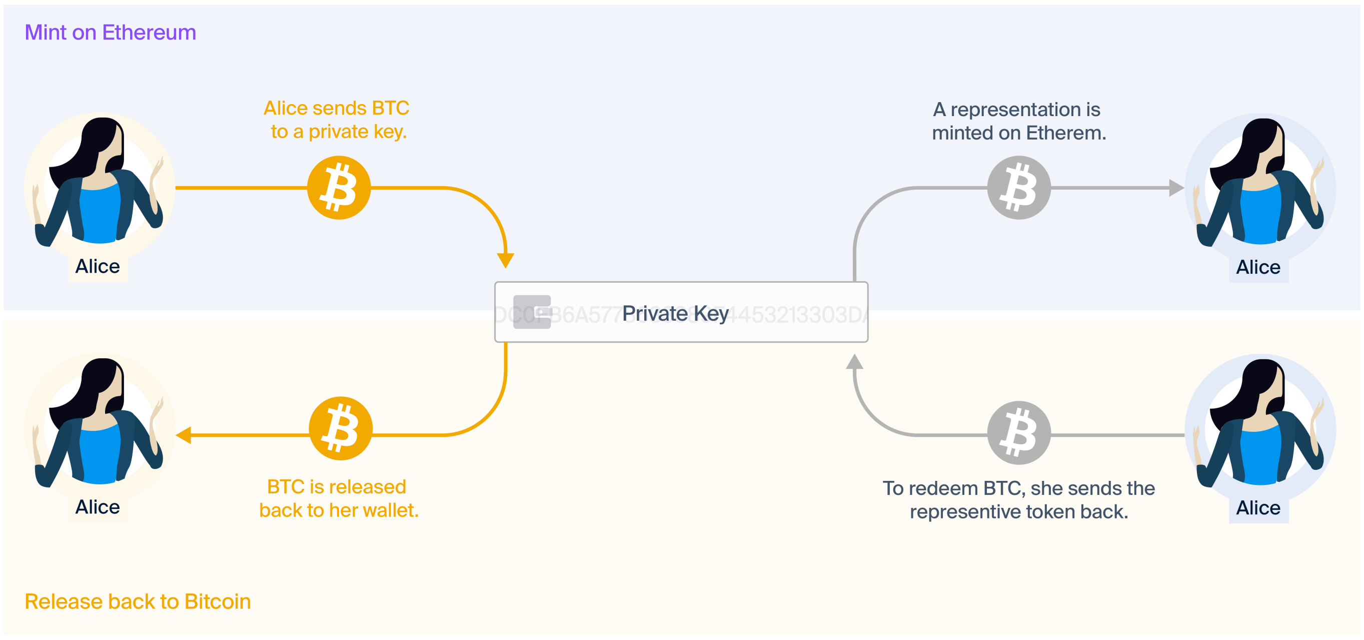


Atomic swaps — or, as they are sometimes miscalled, HTLCs — use special Bitcoin scripts and Ethereum contracts to guarantee that BTC is swapped for ETH/ERC20s in full or not at all. Consider Alice trying to swap BTC for ETH with Bob. Alice will not get custody of the ETH, unless Bob is able to get custody of the BTC, and vice versus. Although atomic swaps have many desirable properties, they have two major drawbacks:

1. They are not universally applicable. Atomic swaps are only usable for swapping, and Alice and Bob must already agree on the assets and the price-point. This makes them very limited in where they can be used. We cannot use atomic swaps to create cross-chain collateralised derivatives, automated market-makers, etc., so other solutions are needed. This problem is particularly apparent when we realise that we want to support applications that may not even exist today, so we need solutions that are as general as possible.
2. They suffer from the free-option problem. Atomic swaps require long timeouts to function correctly. Alice or Bob could intentionally participate slowly, observing market conditions to see if the swap continues to be favourable. A market movement will always make the swap unfavourable for one party, and that party can then cancel the swap. This gives the parties the "option" to back out of a deal that becomes unfavourable. Alice and Bob are both strongly incentivised to behave this way, especially for large amounts, so other incentives (e.g. reputation) need to be brought into the equation.

Synthetics are another form of interoperability that aim to give users exposure to the price of an underlying asset. For example, Dai is a synthetic that gives users exposure to USD. Synthetics generally require the user to deposit an excess amount of collateral to mint a smaller amount of synthetics (e.g. every $150 of collateral allows the minting of $100 of synthetics). If the value of the collateral drops too much with respect to the value of the synthetic, then the collateral is *liquidated*. This means it is taken from the user and used to buy-back-and-burn the synthetic asset that was minted. While synthetics are powerful financial tools, they have major problems when it comes to interoperability:

1. Synthetics are not redeemable for the underlying asset, they are only pegged to be approximately the same price. If you have synthetic BTC, but you now want real BTC, you need to find a counter-party that is willing to make that trade with you.
2. Synthetics cannot interact with other chains. A synthetic that has been minted on one chain can only interact with contracts and assets on that chain, unless it is combined with a different interoperability solution. For example, Dai cannot be moved from Ethereum unless it does so on the back of an interoperability solution like RenVM.
3. Liquidation mechanisms have been known to fail during times of high market volatility. This is problematic, because times of high market volatility are exactly the times when you want your assets to be the most stable/usable. Mass liquidations and rapid price movements can result in synthetic assets that are under-collateralised, and this unpegs their price.


Mint on Ethereum
Alice sends BTC to a private key.
A representation is minted on Etherem.
Alice
Private Key
Alice
Alice
BTC is released back to her wallet.
To redeem BTC, she sends the representive token back.
Alice
Release back to Bitcoin

Lastly, we will look at *tokenised representation*. Tokenised representations are the most flexible kind of interoperability, and can be implemented in many different ways. For example, there is WBTC, imBTC, TBTC, and pBTC. Tokenised representation is where the user locks up an asset with a custodian, and the custodian mints a one-to-one backed token for the user on another chain. This token can then be burned, and the custodian releases the respective amount of the locked asset back to the user. While flexible, all of the existing tokenised representation models exhibit serious problems for universal interoperability:

1. WBTC and imBTC both trust centralised custodians to keep the locked assets secure. While there are many valid use-cases for WBTC and imBTC, they are not decentralised, permission-less, or trust-less. WBTC also enforces that only authorised merchants can request minting/burning, making it impossible for users to directly create/redeem WBTC.
2. TBTC requires synthetic-like over-collateralisation and liquidation. This means the one-to-one peg can be broken by market volatility, and the signers that power the network must accept a lot of risk for little ROI (compared to other investments). It only supports fixed lot sizes of BTC, and requires multiple transactions on both chains, making it overly restrictive for users.
3. pBTC assumes that trusted execution environments are secure enough to resist the attacks of rational adversaries. In practice, many vulnerabilities have recently been discovered that subvert these security assumptions. If pBTC was to lock large amounts of BTC, the incentive to advance and exploit these vulnerabilities would be massive.

RenVM

RenVM implements universal interoperability using the *tokenised representation* model. However, it introduces several advances that solve many of the technical and economic problems in existing models (see above).

RenVM replaces the role of the trusted custodian with a decentralised custodian. This decentralised custodian is implemented using the *RZL MPC algorithm*, which can generate and manage ECDSA private keys without ever exposing them (not even to the machines that power RenVM). This improves on WBTC and imBTC by removing the need to trust a centralised custodian.

RenVM uses *bonding* and *algorithmically adjusted fees* to make sure that attacks are never profitable and to make sure that it can always restore the one-to-one peg if an attack ever does succeed. This improves on TBTC by removing the need for liquidation, which can cause a permanent loss of the one-to-one peg during times of high market volatility. It also improves on pBTC by not relying on trusted execution environments, which have been shown to be exploitable. This approach also allows RenVM to scale its capacity to meet demand: as more assets are locked in RenVM, the algorithmic adjustment of fees allows RenVM to automatically increase its capacity for more locked assets. This is an improvement over TBTC, which requires its signers to explicitly acquire and bond more collateral to increase its capacity.

Lastly, RenVM is designed with careful consideration for the user experience. It allows the minting/burning of pegged assets by anyone, at any time, and at any quantity. Minting/burning of the pegged assets only ever requires *at most one transaction* from the user and can have application-data attached to allow the direct calling of smart contracts. This allows for some interesting use-cases, where users never need to interact with the minting/burning process (and only ever interact with real assets on their real chains) and where cross-chain transactions can be combined/composed to span multiple applications and multiple chains.

How It Works

Although RenVM is capable of supporting complex and composable cross-chain transactions, its design is relatively simple. Here, we will present a high-level overview of how RenVM works, but we will also detail each component in later sections (each component deserves its own dedicated Wiki page).

Darknodes

RenVM is powered by thousands of independently operated machines, known as *Darknodes*, which require bonds of 100K REN tokens to run. The bond of every Darknode represents a commitment to good behaviour and can be slashed if 1) the Darknode behaves maliciously or 2) if it is responsible for the loss of assets (and the slashed bonds can then be used to restore the lost assets).

Shards

Darknodes are periodically shuffled into random non-overlapping groups, known as *shards*. Each shard uses the RZL MPC algorithm to generate a secret ECDSA private key, unknown to everyone, including the Darknodes in the shard. This secret ECDSA private key cannot be revealed and cannot be used to sign transactions, without cooperation of 1/3rd+ of the Darknodes. This enables each shard to securely lock assets into its custody.

Shards are large, containing at least one hundred Darknodes, and they are randomly shuffled once per day. This makes Sybil attacks difficult, as an attacker needs to own a large portion of the entire network to have a chance at corrupting any one shard. This also makes bribery attacks extremely difficult, requiring an attacker to collude with a large number of anonymous Darknodes in a short period of time, with minimal trust.

These properties help RenVM to resist attacks made by irrational adversaries (adversaries that do not care about profiting from an attack). But, it also helps RenVM to resist attacks from rational adversaries during periods where an attack may be temporarily profitable. Regardless, RenVM is always able to restore its one-to-one peg in the unlikely event that an attack succeeds.

Fees

Fees are the main incentive for Darknodes to power RenVM. In return for their work, Darknodes are rewarded with fees that are paid by the user. If the user transfers BTC from one chain to another, the Darknodes earn a small adjustable percentage of that transfer. That is, if BTC is moved by users, BTC is earned by Darknodes, and so on. This helps keep the rewards diverse, and the user experience simple (the user does not need to juggle fee tokens).

Fees are algorithmically adjusted in response to demand. Since REN is only used for bonding, RenVM can use a discounted cash flow model to adjust fees such that the total value of REN bonded by Darknodes is always greater than the total value of assets locked in RenVM. This means that if assets are ever stolen, RenVM can slash the bonds of the responsible Darknodes and use the bonds to restore the one-to-one peg by buying-back-and-burning the same amount of pegged assets. Even if the bonded value temporarily drops below the locked value, RenVM can adjust fees to bring the values back into alignment.

RenVM targets a bonded value that is 3x greater than the locked value, because above this threshold it is irrational to attack RenVM (the loss of the bond is greater than the gain of the attack). However, this is not a hard limit, because as long as the bonded value is greater than the locked value, RenVM can still restore the peg using its buy-back-and-burn mechanism. Furthermore, this mechanism does not need to be applied until an attack is successful, which allows for a time lag between fee adjustment and bond re-evaluation.

Cross-chain Transactions

RenVM supports three kinds of cross-chain transactions. Using BTC-on-Ethereum as an example, these three kinds of cross-chain transactions enable:

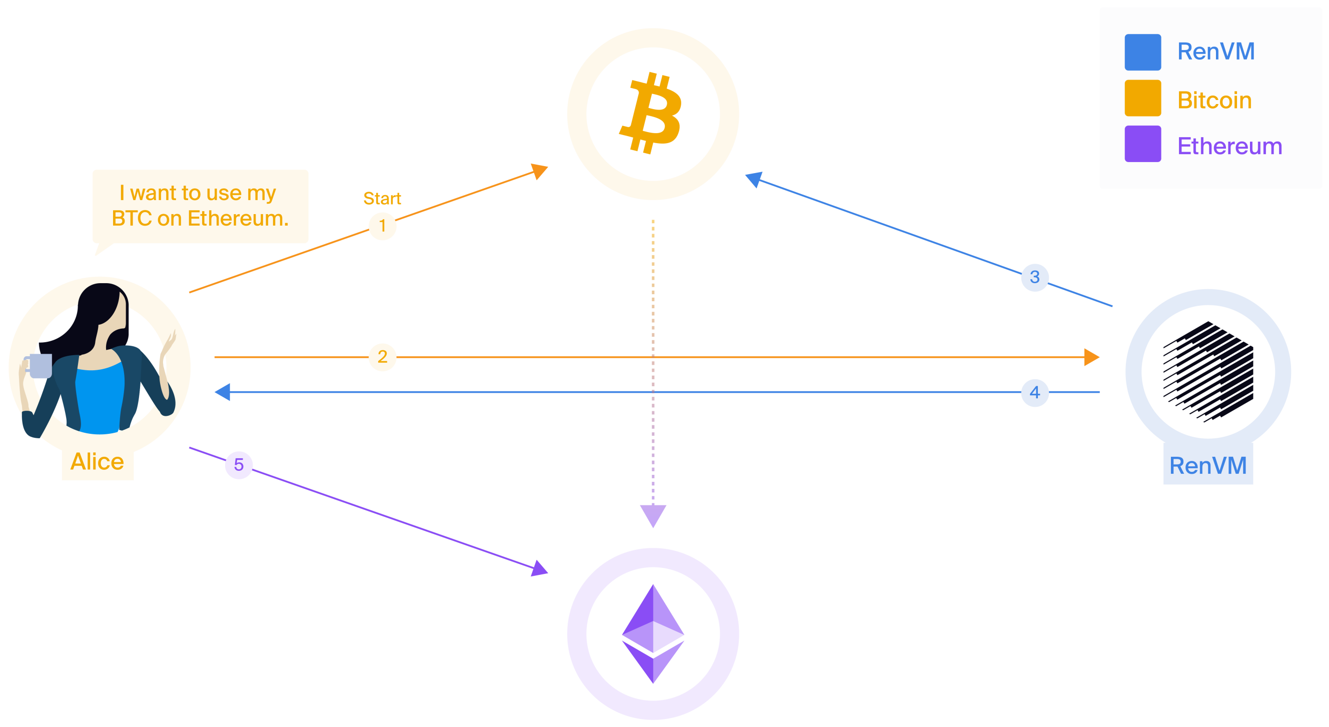
1. sending BTC from Bitcoin to Ethereum (known as a lock-and-mint),
2. sending BTC from Ethereum back to Bitcoin (known as a burn-and-release), and
3. sending BTC from Ethereum to Polkadot (known as a burn-and-mint).

Lock and Mint

Lock-and-mint transactions are cross-chain transactions where the first step, initiated by the user, sends an asset from its origin chain to a host chain. For example, sending BTC from Bitcoin to Ethereum is a lock-and-mint transaction.

Lock-and-mint transactions are so named because the first step requires the user to send assets to RenVM, thereby “locking” into its custody. Unless there is consensus in RenVM that the assets can be released, they will remain locked. After witnessing the locking of assets, RenVM returns a “minting signature” to the user. This authorises the user to mint a tokenised representation of the asset on the host chain. This representation is pegged one-to-one with the locked asset; it is always redeemable in any quantity at any time.

For example, Alice can lock BTC into RenVM, and then mint the same amount of renBTC on Ethereum. She can also attach arbitrary application-specific data, but we will talk about this in more detail later.



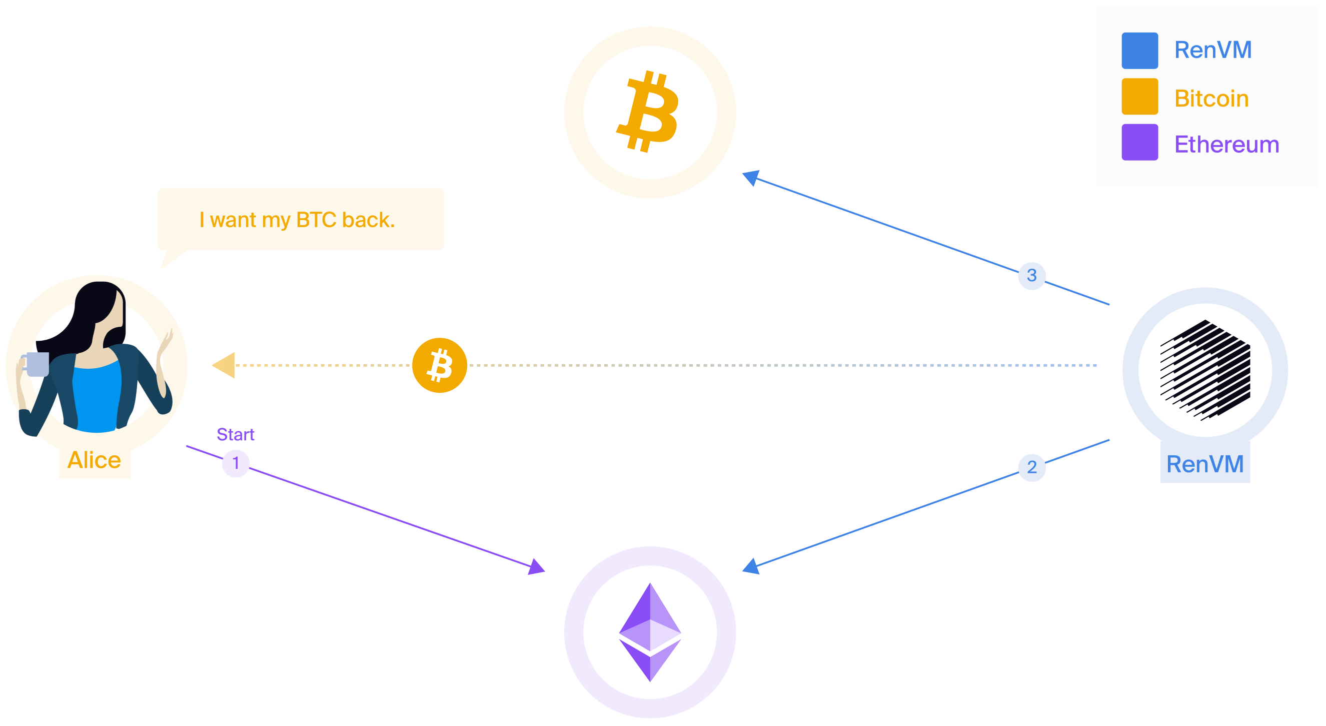
1. Alice makes a Bitcoin transaction that locks 0.55 BTC into the custody of RenVM.
2. Alice (or the application) notifies RenVM about this transaction.
3. RenVM verifies the existence, details, and number of confirmations of the Bitcoin transaction.
4. RenVM uses the RZL MPC algorithm to produce and return a minting signature to Alice.
5. Alice (or the application) submits the minting signature to Ethereum and mints 0.54940005 renBTC (0.55 BTC - fees).

As you can see, only one transaction — the initial bitcoin transaction in the first step — is required from Alice. Everything else can be handled by third-parties. Although it is not discussed here, Alice can attach application-specific data to her cross-chain transaction and the final step can result in smart contracts being called. It is this ability to directly call smart contracts in the final step that allows third-parties, such as the Gas Station Network, to submit transactions of her behalf. Alice never needs to have ETH to pay for gas or even an Ethereum address of her own.

Burn and Release

Burn-and-release transactions are the complement to lock-and-mint transactions and allow users and smart contracts to send assets from a host chain back to their origin chain. The first step, initiated by a user or smart contract, burns the pegged asset from the host chain and specifies an address to which it wants to receive the underlying assets on the origin chain. For example, sending BTC from Ethereum back to Bitcoin is a burn-and-release transaction.

Unsurprisingly, we call such transactions burn-and-release transactions, because the host chain "burns" the pegged assets, and after witnessing the burn, RenVM "releases" the same amount of assets on the origin chain. The burn event specifies the receiving address, which can allow for some interesting compositions of transaction that we will explore later.



1. Alice (or a smart contract) burns 0.2 renBTC on Ethereum, specifying her Bitcoin address at the same time.
2. RenVM witnesses the burn event and waits for the required number of confirmations. RenVM does not need to be notified; it will see the burn event by itself.

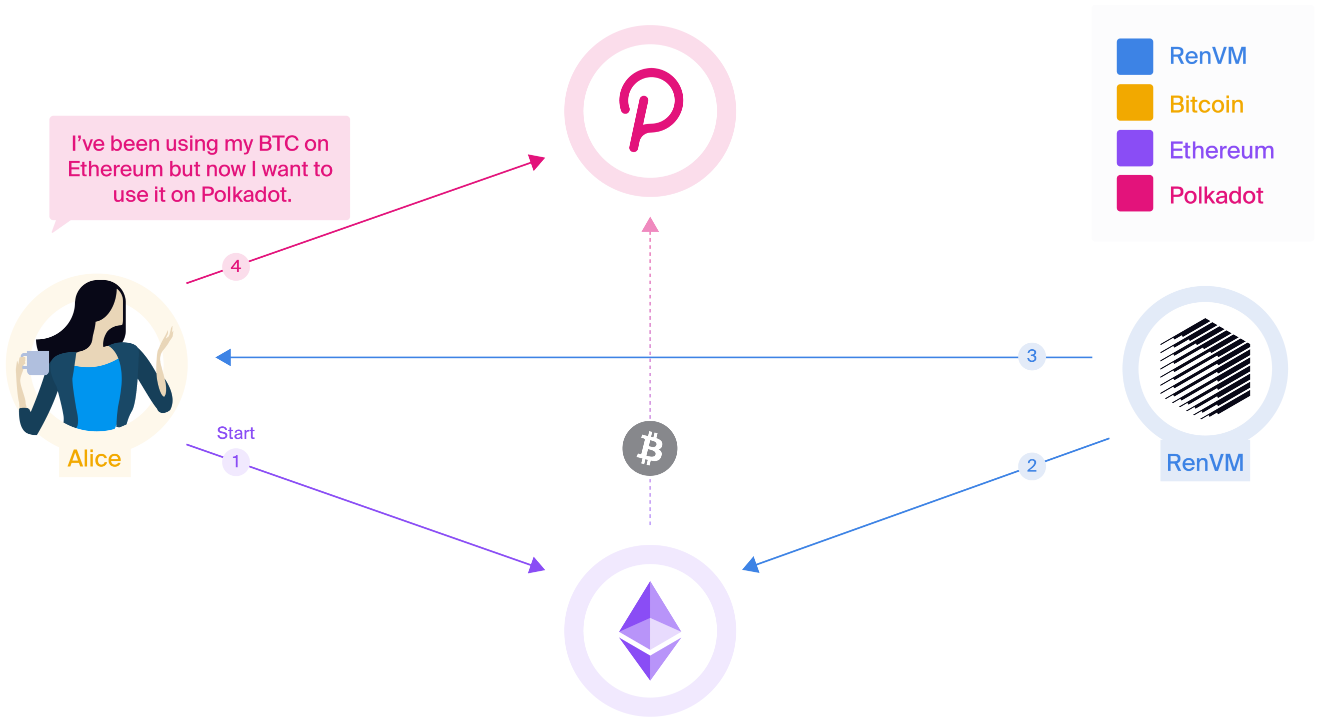
3. RenVM produces a signature that transfers 0.19975 BTC (0.2 renBTC - fees) to the Bitcoin address specified by Alice in the first step.

As with lock-and-mint transactions, Alice is only requires to initiate one transaction in the first step. Everything else is handled by RenVM. The initial burn transaction can also be triggered by a smart contract. In this way, just like with lock-and-mint transactions, third-parties, like Gas Station Network, are able to make the transaction on her behalf (she never needs ETH for gas).

Burn and Mint

Using only lock-and-mint and burn-and-release transactions, we can compose interesting and flexible transactions. One thing we can do is use a burn-and-release transaction to fulfil a lock-and-mint transaction. In effect, this allows us to move an asset from one host chain to another host chain. But, this requires multiple round trips to RenVM, which is unnecessarily expensive and slow (due to underlying blockchain fees and confirmation times).

To better support this kind of transaction flow, RenVM supports *burn-and-mint transactions*, which allow this behaviour in a more direct fashion. Using burn-and-mint transactions, users and smart contracts can “burn” pegged assets from one host chain and “mint” the same amount of pegged assets on another host chain without ever touching the origin chain. For example, sending BTC from Ethereum to Polkadot can be done using a burn-and-mint transaction.


I've been using my BTC on Ethereum but now I want to use it on Polkadot.
RenVM
Bitcoin
Ethereum
Polkadot
4
3
Start
1
2
Alice
RenVM

1. Alice burns 0.34 renBTC on Ethereum, specifying that she wants to send it to her address on Polkadot.

2. RenVM witnessed the burn event and waits for the required number of confirmations. RenVM does not need to be notified; it will see the burn event by itself.
3. RenVM uses the RZL MPC algorithm to produce and return a minting signature to Alice.
4. Alice submits the minting signature to Polkadot and mints 0.33932034 renBTC (0.34 renBTC - fees).

Community

Ask questions, give us feedback, and learn more about the project:

- GitHub
- Telegram
- Twitter
- Reddit

Website | Telegram | Twitter | Reddit

Pages 19

Clone this wiki locally

```
https://github.com/renproject/ren.wiki.git
```

Flexa Network Whitepaper

May 2019 • flexa.network

flexa

At Flexa, we believe that the best way for global commerce to become more efficient and accessible is by making cryptocurrency spendable everywhere.

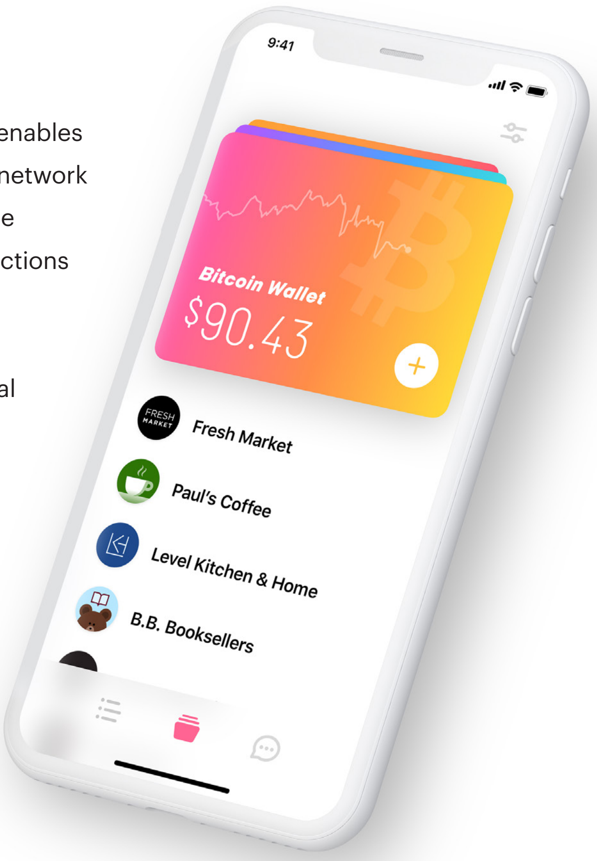
With cryptocurrency transactions exceeding 20 billion USD each day,[1] it's simply a matter of time before digital commodities become a central part of global commerce. And yet, cryptocurrency's collective value of 0.25–0.5 trillion USD[2] remains practically unusable in physical retail.

Considering that 90.9 percent of retail sales in the US still take place offline,[3] brick-and-mortar payments are the primary hurdle in realizing the true utility of cryptocurrencies. Furthermore, widespread retail acceptance of cryptocurrency is critical for its sustainable value.

The solution to blockchain payments

We have developed the Flexa network as an open standard that enables instant cryptocurrency payments in stores and online. This new network is designed to act as an intermediary between merchants and the blockchain, offering them inexpensive and fraud-resistant transactions without volatility exposure. Flexa enables consumers to pay with their preferred cryptocurrency while preserving their freedom, security, and data privacy. And, Flexa doesn't require any physical cards or merchant point-of-sale upgrades.

Flexa was developed from decades of experience in fintech, retail, and payments. Today, Flexa features many high-profile merchants on its platform, the launch of which marks the first real instance of a decentralized, global payment network with the power to make commerce more efficient and accessible for billions of people.



4

Vision

We're making cryptocurrency useful

In the original Bitcoin whitepaper, Satoshi Nakamoto outlined a perspective on the fallacies of modern-day commerce, which relies "almost exclusively on financial institutions... to process electronic payments."[4]

Digital payment instruments in the United States and around the world consist of complicated financial settlement processes—costing merchants *up to 4 percent in processing fees* for purchases and involving up to twelve different entities (each a discrete point of failure) to process a single exchange. Meanwhile, retail fraud losses in the US alone continue to reach all-time highs, claiming more than 48.9 billion USD in 2016.[5]

However, present-day payment instruments are extremely useful to consumers because they have widespread merchant acceptance. And in order for cryptocurrencies to realize similar real and sustainable value, it is critical that they become spendable everywhere.

Many companies have recently developed wallets and apps that enable retail blockchain payments, but they are *universally dependent on existing payment networks*. The promise of cryptocurrency is not being realized when it also requires physical debit cards, linked accounts, or centralized payment infrastructure to facilitate the purchase of a cup of coffee.

Flexa is the first network specifically designed to facilitate practical cryptocurrency payments by enabling instant, no-fee transactions at stores, restaurants, and online. Flexa represents a milestone in the utility of cryptocurrency—payments that are both consumer and merchant friendly. With a simple SDK, Flexa allows developers to add retail payment features to any app, streamlining acceptance of cryptocurrencies for merchants and eliminating volatility exposure.

Flexacoin (FXC) is the new digital collateral token for facilitating retail cryptocurrency payments on the Flexa network. Flexacoin is staked to collateralize every payment on the Flexa network, enabling instant, fraud-free point-of-sale transactions at merchants worldwide—helping to achieve a long-term vision of making cryptocurrency spendable everywhere.

As a simple, neutral, fixed-supply ERC20 token, Flexacoin ensures that the network itself is blockchain-agnostic, and allows people to spend the cryptocurrencies that are meaningful to them. Anyone can use Flexacoin to stake wallets on the Flexa network. Stakers help to collateralize payments made by those wallets, and in return, they earn stake rewards based on transaction volume.

By connecting merchants, banks, and the blockchain with this open network, we’re building a new, global payment system to challenge the status quo.

We envision that the Flexa network will ultimately come to represent open network infrastructure for any blockchain payment, similar to how card associations such as Visa, Mastercard, UnionPay, and American Express offer closed payment rails for credit cards. Beyond that, with digital global payments *in excess of 10 trillion USD each day*,[6] this retail platform will make cryptocurrency more valuable, meaningful, and useful.

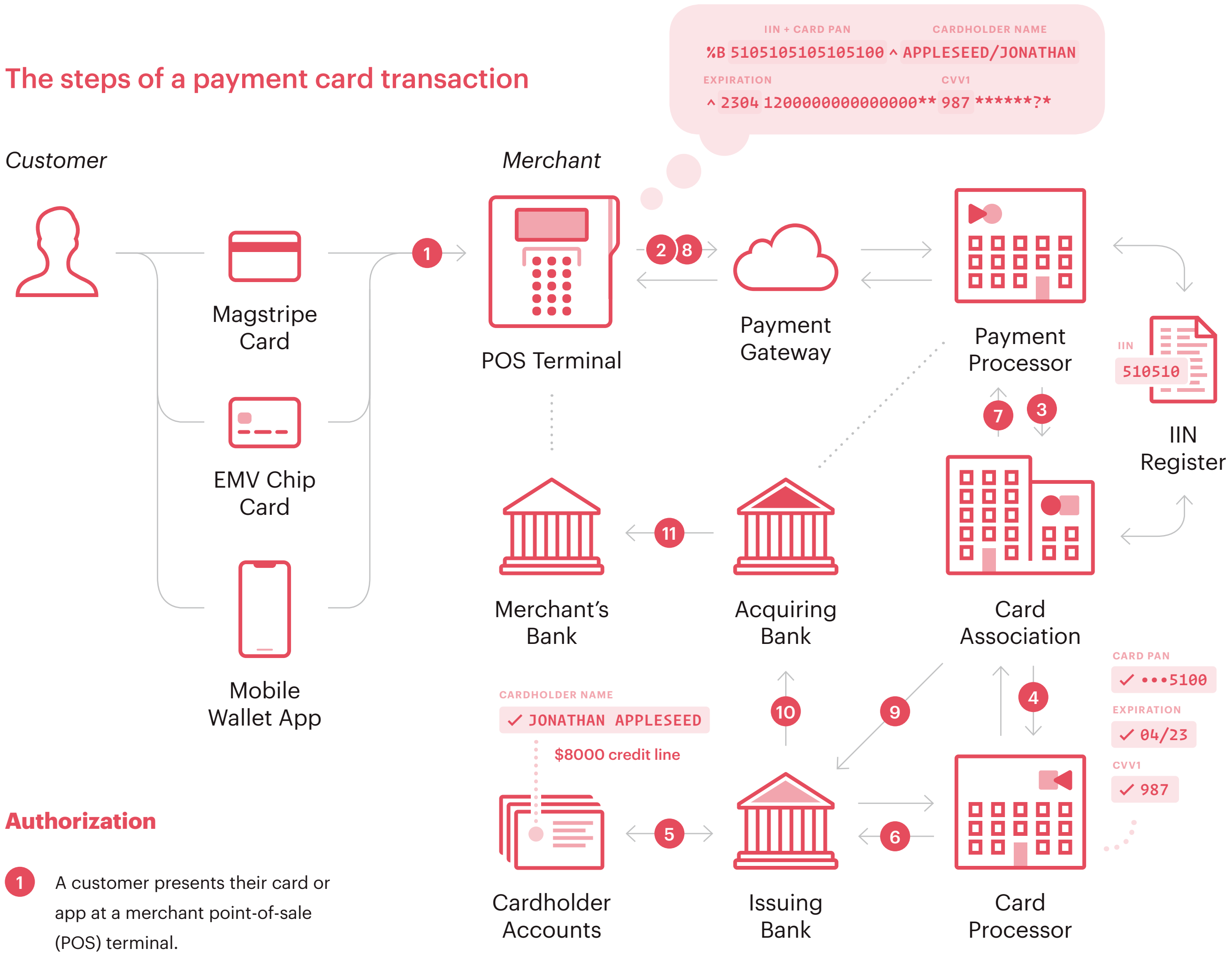
The blockchain as the future of commerce

The limitations of traditional payment instruments—fraud and cost—are solved by the primary strengths of blockchain technology. Accordingly, merchants and the greater blockchain community each stand to benefit from making cryptocurrency spendable everywhere.

For many merchants, payment card fraud and transaction expense are two of the most significant operating costs to manage and actively reduce (e.g., in 2017, losses due to payment card fraud amounted to an estimated 28 billion USD worldwide).[7] Payment card fraud today takes many forms, from stolen account numbers to abuse of marketing incentives. Even chargebacks, initially developed as a consumer protection over forty years ago, have become a vehicle for malicious activity. And smaller merchants ultimately share a disproportionate share of the damages, as they have fewer resources to counter sophisticated fraud or defend themselves in the case of a dispute.

In addition to the costs of fraud, the very act of processing a payment can be extremely expensive, due to the variety of fees and operating expenses involved in handling cash, payment cards, and other payment instruments. For instance, in 2016, the top twenty-five merchants by revenue worldwide spent a collective 19 billion USD to accept payments.[8] In general, these expenses are a result of complex settlement processes across a variety of network participants, including payment gateways, processors, card associations, and financial institutions. Due to this complexity, a standard payment card transaction in the United States involves more than ten discrete steps.

The steps of a payment card transaction



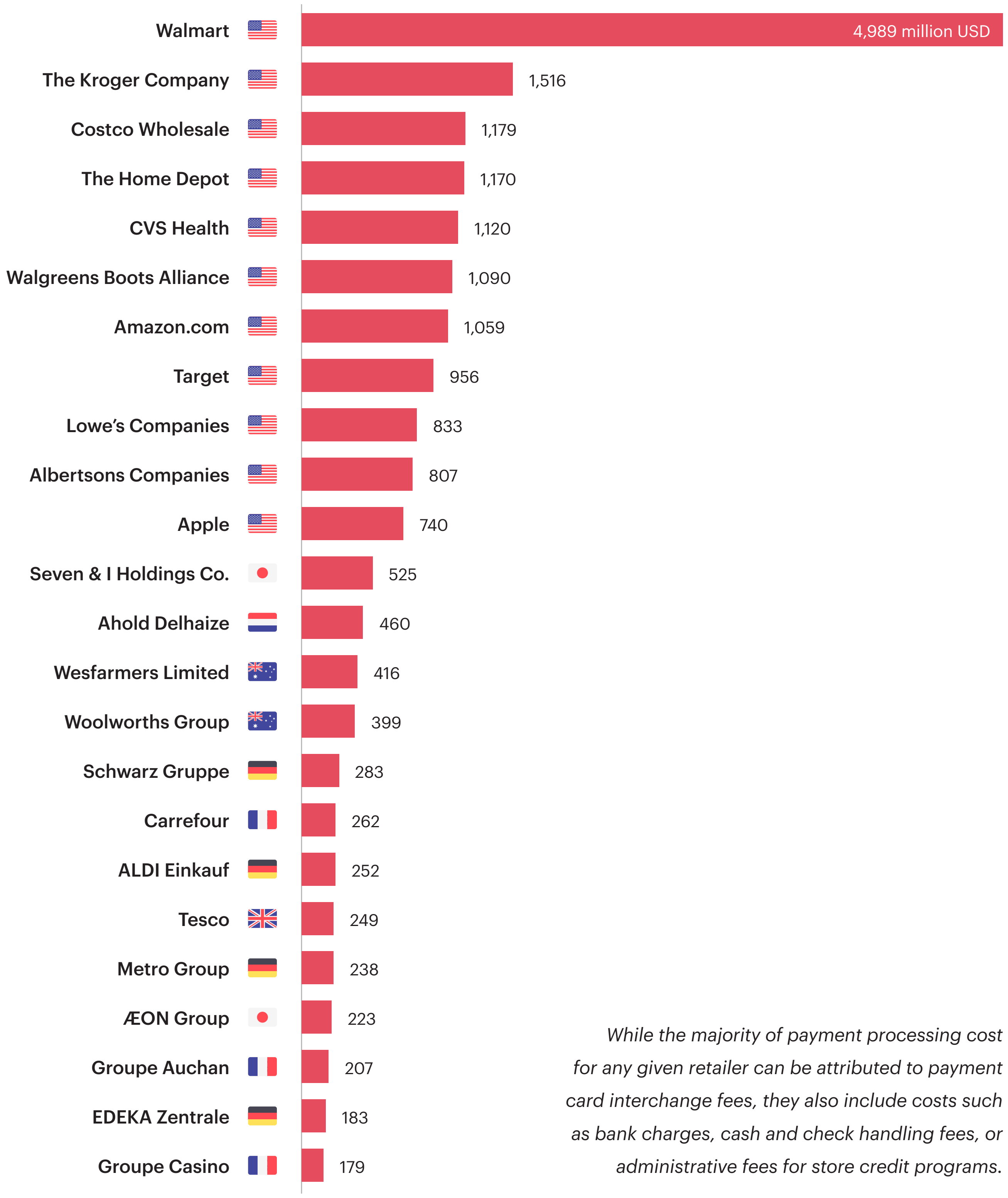
Authorization

1. A customer presents their card or app at a merchant point-of-sale (POS) terminal.
2. The terminal reads the magnetic stripe or embedded signature data from the card and transmits it through a payment gateway to a payment processor.
3. The processor uses a list of Issuer Identification Numbers (IINs) to route data through the appropriate card association, or network.
4. The card association sends the transaction to the bank that issued the card through a card processor.
5. The issuing bank reviews the transaction data, metadata, and internal risk models to determine whether the transaction should be authorized.
6. The issuing bank returns an approval or decline to the card association, along with any other verification data as requested by the merchant.
7. The card association relays the authorization to the processor, which sends a transaction success message back to the POS terminal.
8. Based on the merchant's decision to complete the transaction, the POS terminal sends the payment processor instructions to "settle" the prior authorization amount, which are then relayed to the card association.

Clearing & settlement

9. The card association directs the issuing bank to transfer a final purchase amount (minus interchange) to the processor's own bank, called the "acquiring bank." It returns a success message to the payment processor.
10. The acquiring bank receives funds within 2 business days. Meanwhile, the issuing bank resolves the customer's pending record of charge, and appends it to their statement.
11. The acquiring bank initiates a daily transfer for funds collected minus any fees for processing.

Payment processing costs of the world's largest retailers by revenue in 2016[9]



While the majority of payment processing cost for any given retailer can be attributed to payment card interchange fees, they also include costs such as bank charges, cash and check handling fees, or administrative fees for store credit programs.

How merchants and the blockchain stand to benefit

The blockchain offers a practical solution to merchant concerns of fraud and cost. It dramatically reduces the number of possible fraud vectors by enforcing tamper-proof transactions on a ledger, and it decentralizes transaction verification—creating an open market for processing that more closely represents the actual computation cost. As an added benefit, the blockchain provides native support for borderless payments, which opens merchants to a global community of customers without requiring additional payment infrastructure or currency exchange.

Meanwhile, the blockchain needs merchant adoption in order to become a viable supplement to other payment methods. Additionally, with this increased utility comes more straightforward cryptocurrency valuation, reduced volatility, and market stabilization. Growing merchant acceptance will make cryptocurrencies substantially more valuable, and truly enable the globalizing effects of peer-to-peer electronic cash that Satoshi Nakamoto envisioned.

Blockchain adoption is inevitable

For these reasons, we believe that the blockchain complements the infrastructure of traditional payment instruments. However, due to the operational and technical complexity in managing native cryptocurrencies at scale, many merchants will require an intermediary service. This service must be designed so that it cannot compromise the core principles of data protection, decentralization, and choice that have bolstered the cryptocurrency community since its inception.

We believe that the Flexa network offers the first practical cryptocurrency payments service for retail, dining, groceries, fuel, travel, and more. We remove the complexities of acceptance to bring fraud resistance and low-fee processing to merchants, while still protecting consumer tenets of privacy, decentralized governance, and freedom of choice. By allowing merchants and their customers to engage directly as buyers and sellers, global commerce becomes vastly more efficient.

A framework for consumer adoption

The software that moves the vast majority of money around the world today still uses legacy standards created during the late 1970s,[10] but in the absence of a compelling alternative, consumers are trapped into maintaining the status quo.

Since the creation of Bitcoin in 2008, blockchain communities have attempted to make cryptocurrencies a useful complement to traditional payment instruments like credit cards, debit cards, and cash. However, fundamental user-experience challenges such as unintuitive QR code interfaces, complex address strings, new security protocols, and network capacity issues have hindered commercial adoption. Various scaling solutions such as multi-layer protocols and Proof of Stake consensus algorithms show considerable promise for improving the speed and utility of blockchain transactions, but create issues of complexity and compatibility for merchants.

A variety of mobile wallets are promoting cryptocurrency payment solutions, but unfortunately, they are completely reliant on existing legacy infrastructure. These wallets utilize high-fee virtual Visa and Mastercard debit cards—requiring bank accounts, physical cards, and multiple tiers of centralization. Digital payments on these platforms are subject to low transaction limits (in some cases, less than $100),[11] as well as Apple's restrictions for NFC access on iOS devices.[12] Justifiably, these systems have extremely low consumer adoption due to the increased friction compared to a typical payment card.

The solution to blockchain payments is not building cryptocurrency acceptance on top of the existing multi-layer networks, but creating a *new network* that solves merchant and consumer needs alike.

While decades of retail payments experience confirm the pain points of fraud and processing cost, we find that consumers' needs are distinctly different. Consumers evaluate payment instruments against an individual framework of five basic criteria:

Basic consumer criteria	*Bank transfers*	*Payment cards*	*Mobile wallets*	***Flexa***
Freedom of choice The need to avoid fees, and mechanisms of unwarranted control	✗	✗	✗	✓
Security The need to use a system without fear of loss by deception or failure	✓	✓	✗	✓
Speed The requisite convenience of instant confirmation, often lost to security	✗	✗	✓	✓
Usefulness The need for widespread acceptance of a particular payment instrument	✓	✓	✓	✓
Value Any incentive to use a payment instrument (e.g., rewards, no fees)	✗	✓	✗	✓

Speed, usefulness, and value are often the most critical factors in choosing a particular payment instrument at retail. Each of these features must be addressed for cryptocurrencies to see widespread adoption.

In order for a viable blockchain cryptocurrency payment network to achieve meaningful scale, the table stakes for consumers are the following:

1 Real-time transactions

Merchants and their customers need to receive confirmation that a transaction was successful in less than one second.

2 No consumer-facing fees

Consumers will not pay a premium to use blockchain cryptocurrencies, because such a cost represents negative value in their decision-making framework. The fee must be zero on the consumer side of the transaction, and ultimately provide competitive spending incentives.

3 Broad acceptance

In order to see widespread consumer adoption, it must be possible to use cryptocurrencies for the majority of daily expenditures. Any less than that, and the mindshare required to maintain "front of wallet" utility will not be attainable.

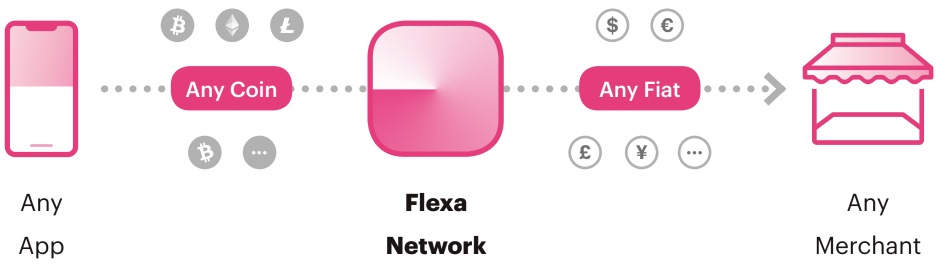
Meeting and dramatically exceeding these expectations will be challenging, but any new payments network must comprehensively solve both consumer and merchant needs. We believe that Flexa satisfies all of the core consumer requirements necessary to break the legacy payments status quo.

Product

Introducing Flexa

Flexa is an open network for enabling instant cryptocurrency payments in stores and online, allowing merchants to receive secure cash deposits via their existing points-of-sale.

Scalable retail payments for any app

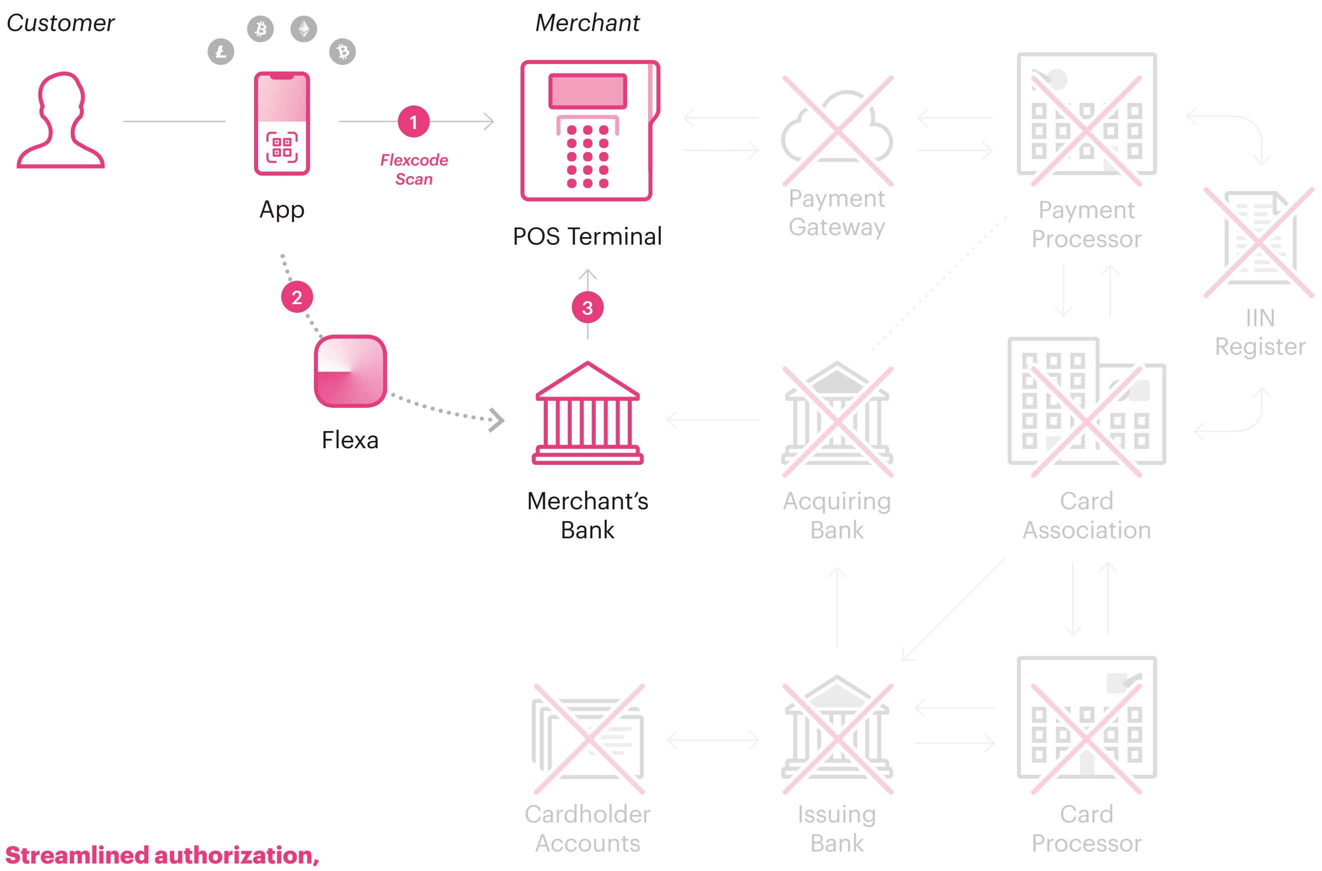


Flexa is designed to facilitate payments from any wallet, in any coin, to any merchant, across the globe. Flexa's network is already integrated with many high-profile merchants, offering instant acceptance of potentially hundreds of cryptocurrencies to developers all over the world.

The vision for this new network is to become the open, seamless standard for cryptocurrency payments in physical retail.

Flexa will enable developers to integrate retail cryptocurrency payments within their own apps. By creating the most simple, direct network, Flexa enables broad cryptocurrency acceptance with the least complexity—no longer requiring the variety of payment gateways, processors, associations, and financial institutions.

Sample Flexa transaction flow



Streamlined authorization, clearing & settlement

1. A customer scans their app at merchant POS for payment in any cryptocurrency supported by Flexa.

2. The app requests the current conversion rate for the customer's desired cryptocurrency, and submits a blockchain transaction via Flexa.

3. The Flexa network transmits a one-time authorization code (FPAN) in real time to authorize the transaction on the merchant's POS terminal, then pushes fiat funds to the merchant's bank account. The customer's purchase is complete.

The Flexa payment experience

Flexa payments are designed to be as simple as possible. With just a single tap and scan, Flexa verifies your cryptocurrency balance against a public index rate and generates a proprietary flexcode for payment.

Because Flexa payments do not require NFC (like traditional payment cards), they are not restricted by Apple's requirements for payment cards to be loaded into the Wallet app, nor by tap-to-pay (contactless) implementation timelines or transaction limits. This greatly reduces the network's risk as compared to other cryptocurrency payment solutions.

A Flexa transaction has two primary components, which are delivered to client apps through the Flexa Wallet SDK:

- The first is called an **FPAN**, or flexible primary account number, which is a one-time authorization that allows a merchant to debit local fiat currency against the selected cryptocurrency wallet balance.

- The second is called a **flexcode**, which is a proprietary and backwards-compatible barcode format that is scannable by standard point-of-sale barcode readers. Each flexcode conveys the FPAN with any user-authorized metadata through the merchant's point-of-sale system.

Because all Flexa transactions use the same authentication process for payments, they represent the only interface that is just as secure—and just as usable—whether used in stores or online.

Online Flexa transactions will make use of identical FPAN provisioning mechanisms and back-end integrations. In fact, online Flexa transactions will differ from physical Flexa transactions only in their form of approval. Instead of using a flexcode, virtual Flexa transactions will relay an FPAN via an account-linked device.

Principles for network development

Flexa has presented six principles for the network, representing our vision for its ongoing development and sustained platform growth.

In order to become a trusted, public cryptocurrency payment rail, we believe that Flexa must be:

Compliant **Secure** **Instant** **Open** **Simple** **Useful**

We present these principles to help guide the development of Flexa, and we hope to build trust and transparency with the blockchain community by articulating them as the ongoing intentions guiding the network:

Compliant

We designed Flexa to support local compliance requirements and data protection regulations. Notably, unlike alternative solutions for institutional cryptocurrency payments, the network does not require Flexa or third parties to act as custodian of funds; the technology functions as a direct payments processor without volatility exposure for merchants.

Secure

The Flexa network has been designed with open-source, end-to-end encryption for resilience from man-in-the-middle attacks and other forms of surveillance or tampering, and exposes only non-sensitive information in the course of completing a transaction.

Instant

Flexa is the only network to offer instantaneous conversion of cryptocurrency via direct bank deposits at merchant point-of-sale, regardless of block time. End users need only one tap to authorize payment, with transactions (confirmed by point-of-sale) currently measured at less than a second. Flexa transactions are designed to be the absolute fastest payment solution available in the world.

Open

Flexa is designed to enable the free and open use of cryptocurrency at retail. The network will be accessible to a wide variety of developers and merchants around the globe. To support this widespread acceptance, the network community need only stake Flexacoin. Therefore, Flexa requires no proprietary license or gateway in order for developers to integrate their wallet or transmit cryptocurrency transactions.

Simple

From tap to transaction, Flexa supports simple, straightforward API methods for exchange and payment. Because the network is not reliant on existing payments infrastructure, payments are pre-authorized in a single message, enabling authorization signatures and settlement to be combined into one fraud-resistant transaction. Flexa's simplicity of integration, operation, and settlement makes cryptocurrency payments easy for merchants and their customers alike.

Useful

Finally, Flexa is designed to be backwards-compatible with existing POS systems, and as interoperable with as many partners and platforms as possible. We have developed the Flexa network toward broad accessibility and widespread acceptance—starting with the very first apps on the network, which take advantage of existing POS integrations and require no new hardware or merchant upgrades.

People

Our team

The people behind Flexa combine more than twenty decades of experience in technology, retail and payments at American Express, Bloomberg, the MIT Media Lab, NASA, Starbucks, and Warby Parker.

Tyler Spalding

Co-Founder

Tyler has founded and invested in various blockchain projects since 2011. He was previously the CTO of Raise, Co-Founder and CTO of Tastebud Technologies, and an Engineering Lead with the United Space Alliance, US Air Force, and NASA's Space Shuttle Program. He holds two Masters degrees from MIT and UIUC.

linkedin.com/in/tylerspalding

Trevor Filter

Co-Founder

Trevor began his career at the MIT Media Lab, and has been designing award-winning, customer-centric experiences for over a decade. He was previously Head of Product & Design at Raise, Head of Product at Slide Network, and a Senior Product Manager at American Express. He holds a Bachelors from MIT.

linkedin.com/in/trevorfilter

Zachary Kilgore

Co-Founder

Zach has more than eight years of experience engineering front-end and back-end software platforms and infrastructures for payments and mobile. He was previously an Engineering Manager at Raise, Director of Engineering at Slide Network and a Front-End Engineer at Warby Parker. He holds a Bachelors from Duke University.

linkedin.com/in/zacharykilgore

Daniel C. McCabe

Co-Founder

Daniel has 20 years of experience across finance, technology, and private equity law. He was formerly a partner at Greensfelder and holds a JD from the Chicago Kent College of Law with a Bachelors from Northwestern University.

linkedin.com/in/danielcmccabe

Ryan Records

VP of Partnerships

Ryan led the creation, rollout, and consumer growth strategies for the Starbucks mobile app, one of the most successful mobile payment platforms in the world. He holds a Masters degree from Washington State University.

linkedin.com/in/ryanrecords

Caitlin Skulley

Sr. Director of Merchant Dev.

Caitlin built and grew the merchant B2B program from the ground up for a leading payments distributor, and boasts nearly 20 years of experience in client services. She holds a Bachelors degree from Colby College.

linkedin.com/in/caitlinskulley

Alex Disney

Blockchain Engineer

Alex is a blockchain engineer with ten years of experience developing cryptocurrency mining and trading operations at DRW. He implemented EIP-758 for Parity and holds a Bachelors degree from UIUC.

linkedin.com/in/alex-disney-4a203617

Chris Pick

Software Engineer

Chris is a financial software engineer with seven years of experience building distributed data storage and analysis systems and infrastructure at Bloomberg. He holds a Bachelors degree from UIUC.

linkedin.com/in/christopherpick

Advisors

To guide the growth and scale of our products, we have also assembled a group of talented individuals across blockchain development, consumer retail, hardware, machine learning, marketing, and payments. Our advisors bring the experience of leadership positions with some of the most notable companies in the world, including:

Amazon.com	**Google**	**Tesla**
American Express	**Mastercard**	**Venmo**
Apple	**Nike**	**Visa**
Capital One	**PayPal**	**Walmart / Store No. 8**
Citigroup	**Pinterest**	**Warby Parker**
ConsenSys	**Samsung**	

Payments, financial services, and blockchain

Luke Gebb

SVP of Amex Digital Labs and Global Network Products

linkedin.com/in/luke-gebb-12812b

Dave Hoover

Ethereum Developer, formerly at ConsenSys and IDEO

linkedin.com/in/redsquirrel

Mark Jamison

Global Head of Innovation at Visa and former CDO at Capital One

linkedin.com/in/markrjamison

Jason Korosec

Former SVP and Group Head of Information Services at Mastercard

linkedin.com/in/jasonkorosec

Pete Woodhouse

Former CTO of PayPal Credit and Sr. Director of Global Solutions

linkedin.com/in/woodhouse

Retail, hardware, and consumer partnerships

Sharat Alankar
Blockchain and Incubation Associate at Walmart Store No. 8
linkedin.com/in/sharatalankar

Shahriar Khushrushahi
Senior Engineer on Google ARA and Project Jacquard, MIT PhD
linkedin.com/in/skhushrushahi

Thomas Kim
Former Board Director at ACTnano, products for Apple, Google, Tesla
linkedin.com/in/thomaskimco

Brian Magida
Director of Performance Marketing at Warby Parker
linkedin.com/in/brian-magida-46186312

Deirdre Peters
Senior Product Manager for e-Commerce at Nike
linkedin.com/in/dierdre-peters-9909318

Chris Walti
Former RFID Lead at Accenture Technology Labs, MIT Media Lab
linkedin.com/in/chriswalti

Christina Wick
Former Head of Engineering at Venmo and Sr. Manager at Amazon
linkedin.com/in/christina-wick-60b4981

Branding and marketing

Coby Berman
COO at Radar, former Sales Director at mParticle, Foursquare
linkedin.com/in/cobyberman

Anthony Rodriguez
Founder and CEO of Emmy-winning agency Lineage Digital
linkedin.com/in/avrod

Sarah Shere
Pinterest Head of Product Marketing, former Sales Manager at Google
linkedin.com/in/sarah-hoople-shere-87a2413

Background

The present state of digital payments

Digital payments take many forms around the world and move a collective 10 trillion USD each day.[13] The majority of these transactions are conducted using one or any combination of three instruments:

Direct bank transfer

e.g., SWIFT, Fedwire, ACH

Global and domestic bank clearing networks that move 3.6 quadrillion USD in 102 million transactions per year[14]

Common throughout Europe for all transaction sizes, and in the US and Canada for large and commercial transactions

→ Page 27

Payment cards

e.g., Visa, American Express

Plastic cards leveraging credit and debit networks to move small purchases of 26 trillion USD in 257 billion transactions per year[15]

Common in most geographies throughout the world for small transactions, especially the US, Canada, Europe, and Asia-Pacific

→ Page 30

Mobile wallets

e.g., Apple Pay, Google Pay, Alipay

Mobile apps that proxy traditional payment instruments to move more than 8 billion USD in 300 million transactions per year[16]

Common in Asia-Pacific for all transactions (via bank transfers); gaining broad acceptance in the US, Canada, and Europe

→ Page 33

Direct bank transfer

In general, non-cash payment instruments are underpinned by a traditional account held at an insured financial institution, such as a commercial bank or credit union. Whenever money is exchanged via one of these payment instruments, whether electronically or by an offline ledger, it is ultimately transmitted between financial institutions. To reconcile these payments, a variety of domestic and international standards are used for direct bank transfer between businesses and consumers (also sometimes called "electronic funds transfer"), such as ACH/IAT, CHIPS, SWIFT, RTGS, Fedwire, BEPS, NEFT, and KFTC.

Despite their ubiquity, each of these systems rely on legacy infrastructure that remains vulnerable to fraud and transaction inefficiencies.

Legacy infrastructure

The underlying technology of the global financial network is difficult to navigate, consisting of a variety of incompatible legacy protocols and standards; many of the current electronic settlement systems have remained relatively unchanged for 40 years. For instance, Automated Clearing House (ACH) transactions in the United States are still conducted via fixed-width text files (with precisely 94 characters per line), uploaded to various FTP servers and downloaded at specific times of day for settlement. Until 2016, these transactions cleared the following business day, when NACHA announced an update allowing for same-

Daily processing volume

SWIFT
5 trillion USD[17]
30.7 million transactions[18]

Fedwire
2.1 trillion USD[21]
528,000 transactions

CHIPS
1.4 trillion USD[21]
430,000 transactions

ACH
120 billion USD[19]
70.1 million transactions

day payments.[20] This “upgrade” involved no changes to the underlying specification; rather, banks were required to process transactions twice instead of once daily.

Other clearing systems include the Society for Worldwide Interbank Financial Telecommunication (SWIFT); the New York Clearing House Association’s CHIPS network; and the Federal Reserve’s Fedwire network. Each involve substantially more robust checks and balances than ACH and benefit from greater speed, increasing the complexity of the global financial system. Together, these systems transmit a staggering 3.6 quadrillion USD in global volume.[21]

Fraud vulnerability

Despite the additional supervision involved in these ledger systems, their protocols and networks are vulnerable to fraud. In a 2016 survey of the largest financial institutions, “cybersecurity concerns” was the most-responded challenge that bank executives said they faced in their day-to-day role,[22] and many such instances of theft have recently become public.

In 2016, thieves made off with 81 million USD by impersonating Central Bank SWIFT operators.[23] Throughout a single weekend, they routed four transactions through the New York Fed’s mostly automated system, moving 101 million USD from Bangladesh to the Philippines. It was only when a New York Fed official caught a thief’s misspelling of the beneficiary name that they were able to alert Bangladesh Bank officials and prevent the transit of an additional 920 million USD.

In 2018, a larger heist was discovered involving the Punjab National Bank and promissory “letters of understanding” issued through SWIFT, where funds were laundered by using a password provided by bank officials for direct access to the SWIFT network.[24] Letters of understanding were issued for the equivalent of nearly 1.77 billion USD, and they were not correlated with the lesser amount that was registered via SWIFT in the bank’s holdings. Despite repeated warnings against fraudulent SWIFT messaging from the deputy governor of the Reserve Bank of India, the scam went undetected for nearly seven years.[25]

Transaction inefficiency

Despite the underlying fraud vectors, funds transmitted over SWIFT, Fedwire, CHIPS, and ACH incur costs of approximately 18 billion USD every day.[26] Additionally, transfers require three to five days for settlement, and up to 4 percent of payments fail due to technical reasons.[27]

The blockchain could potentially offer several enhancements in these systems, namely cryptographically secure transactions, immutability, and data redundancy. For instance, Ripple, a prominent US startup, allows financial institutions to quickly settle cross-border payments using its xCurrent network, claiming a 60 percent reduction in net cost.[28] Remittance providers such as Western Union and Moneygram have also piloted using native Ripple blockchain tokens (XRP) for settlement.[29] Using products such as these, we believe that blockchains have the potential to influence well beyond the primary layer of the global financial network.

Payment cards

Direct bank transfers are just one of the several steps involved in conducting a standard transaction with a payment card (e.g. a credit or debit card). In practice, the payment card authorization and settlement framework implemented throughout the United States and Europe involves the coordination of no fewer than six parties in order to transmit and guarantee funds.

Although payment cards offer universal acceptance and consumer benefits, they are prone to many single points of failure as well as the rising costs of fraud and incentive fees.

Many single points of failure

The companies involved in payment card processing serve mutually exclusive roles and extract a share of the transaction fee. This fee is called "interchange," and has been variously regulated by the European Union (Interchange Fee Regulation, April 2015), and the Federal Reserve (Durbin Amendment, July 2010).

Payment cards also mandate a secondary network provided by entities called "card associations." Card associations work with payment processors to conduct the three broad stages of a payment card transaction: authorization (verifying funds in accounts on either side of a transaction), clearing (transferring funds between banks after the exchange of goods or services) and settlement (paying a merchant).

Daily processing volume

Union Pay
41 billion USD[30]
105 million transactions[31]

Visa
20 billion USD[32]
305 million transactions

Mastercard
12 billion USD[33]
184 million transactions

American Express
3.2 billion USD[37]
19.8 million transactions

JCB
731 million USD[37]
8.1 million transactions

Discover
466 million USD[37]
6.4 million transactions

In order to accept payment cards, merchants incur disproportionately high processing fees which are often one of their largest operational costs.

In 2012, responding to these rising processing fees, some of the largest merchants in the US—including Walmart, Target, Best Buy, CVS and 7-Eleven—created a cooperative organization called Merchant Customer Exchange (MCX), with the charter of developing an ACH-backed payment instrument to avoid interchange fees.[34] After three years of continuous merchant investment and delayed development, the MCX mobile app never exited a pilot phase. Although it was successful at reducing merchant costs of processing, MCX was never able to deliver a compelling consumer value proposition. In 2017, JP Morgan Chase acquired the MCX technology to integrate with its existing Chase Pay system.

In the past decade, payment card processing fees have skyrocketed for two main reasons: first, because of an increase in fraud, including losses that are paid by issuing banks when they reimburse their customers for unauthorized charges; and second, because of the consumer demand for better card benefits and rewards on high-end credit card products.

Rising costs of fraud

EMV (Europay Mastercard Visa) chip cards have found mainstream adoption in Europe, Asia and the US, but payment card fraud in aggregate has continued to rise. Despite broad acceptance of the card-based technology, 2.8 million fraudulent accounts were created in 2018,[35] and account takeovers cost merchants 5.1 billion USD.[36] Additionally, transactions made online (i.e., “card-not-present”) have seen fraud losses increase more than 100 percent since the introduction of the EMV standard.[37] Chip-enabled cards have subsequently increased payment security, but are still vulnerable to man-in-the-middle attacks, especially when merchants don’t upgrade their systems to support encrypted transaction data from EMV-capable terminals. Cards can also be cloned from unsophisticated account enumeration, physical card skimmers, RFID readers, or simply a restaurant waiter with a cell phone camera.

In 2016, the total fraud losses for payment cards worldwide was estimated at 22.80 billion USD,[38] with 46 percent of all US citizens reporting card fraud within the past five years.[39]

Recently, the rate of identity theft has soared, with more than 1,500 corporate data breaches,[40] including the theft of 143 million credit reports from Equifax[41] and 40 million credit card numbers from Target.[42]

Yahoo—now part of Verizon—also revealed that hackers obtained the personal information of its entire database of 3 billion worldwide users during an attack in 2013.[43]

Incentive fees

Credit card rewards points also contribute to the high fees incurred by merchants to accept these payment instruments. Originally introduced by American Express in 1991,[44] these points have since become a cornerstone of consumer marketing for major credit card products. Today, travel and dining bonuses have become extremely competitive for the major credit card issuers: Chase, Capital One, and American Express are each vying for coveted “front of wallet” placement by offering up to 5× points or five percent cash back on various purchase categories.

As a result, many industries have developed to help the affluent consumer “optimize” their spend for maximum returns, perhaps without realizing that the true cost of these rewards is subsidized either by the small merchant businesses (which lack the required leverage to negotiate more affordable interchange rates), or the other payment card consumers who finance debt through high monthly APR interest. Many small businesses ultimately choose to avoid payment cards altogether and revert to cash-only transactions, putting them at a significant consumer disadvantage.

Mobile wallets

More and more, third-party mobile wallets are becoming mainstream payment instruments, capitalizing on their ability to aggregate various aspects of bank accounts and payment cards and offer even more consumer choice and convenience. While some (like Apple Pay, Google Pay, and Samsung Pay) simply serve as vehicles for virtual cards by proxying existing payment cards' primary account numbers, or PANs; others (such as Alipay, WeChat Pay, PayPal, Venmo, Square Cash, and Apple Pay Cash) have built a suite of value-added services and integrations on top of what is essentially a stored value account.

Many of these mobile wallets have seen substantial growth in recent years—especially in China—but their traditional payment instrument underpinnings present limitations on the ability to provide meaningful incentives, grow internationally, and manage fraud vulnerabilities.

Limited incentives

Today, even the largest and most successful mobile wallet apps and services enable the vast majority of their transactions by proxying an underlying insured or regulated payment instrument, such as a bank account or payment card. By functioning as this abstraction layer, services like Apple Pay and PayPal are able to offer value-added features like enhanced security or purchase protection, but are limited in their ability to provide unique incentives or sustainable bonus structures beyond what the underlying instruments already support natively.

Daily processing volume

Alipay
4.7 billion USD[45]
175 million transactions

WeChat Pay
3.3 billion USD[46]
130 million transactions[47]

PayPal (incl. Venmo)
425 million USD[48]
8.3 million transactions[49]

Paytm
55 million USD[50]
11 million transactions

International incompatibility

Moreover, mobile wallets have seen substantial growth in markets without entrenched financial institutions. For example, in the absence of traditional, credit-based payment infrastructure throughout China, companies like Alipay and WeChat Pay have built a direct system that facilitates mobile transactions on a private payment network over the internet. The rapid growth of these platforms—in terms of both scale and versatility—is impressive. But because the underlying financial infrastructure is still provided by domestic financial institutions, international growth is encumbered by the overhead of adapting these systems to foreign banks and exchanging currencies.

Outside of payments, the major value in third-party mobile wallets is their usefulness for internal or peer-to-peer transactions via network effects. Many people join Alipay, WeChat, Venmo or Square Cash because their friends are there, or because it's easier to send money to a phone number or username than it is to share account numbers. These features build community, but ultimately limit platform growth to these regional groups because users have limited incentive to interact internationally.

Fraud vulnerability

Mobile wallets are essentially an interface to existing payment instruments, which can make them vulnerable to certain types of fraud. By storing many payment instruments behind a single online account and password, these apps create an opportunity for account takeovers, which in 2018 amounted to 5.1 billion USD in losses.[51] Many apps also distinguish between peer-to-peer payments and payments for goods and services because of their inability to mitigate buyer fraud, such as chargebacks and ACH returns.

For example, due to its ACH underpinnings, Venmo's terms and conditions explicitly warn against using the app for retail payments. When a fraudster reverses an ACH transaction used to load a Venmo account, the company is forced to reverse the transaction within its own ecosystem, sometimes by directly debiting beneficiaries' bank accounts.

References

1. "Cryptocurrency Market Capitalizations," CoinMarketCap, April 30, 2018, https://coinmarketcap.com/charts/.

2. CoinMarketCap.

3. "Quarterly Retail E-Commerce Sales," U.S. Census Bureau, PDF, February 16, 2018, https://www.census.gov/retail/mrts/www/data/pdf/ec_current.pdf.

4. Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," PDF, October 31, 2008, http://bitcoin.org/bitcoin.pdf.

5. National Retail Federation, "National Retail Security Survey 2017," PDF, June 22, 2017, https://nrf.com/system/tdf/Documents/NRSS-Industry-Research-Survey-2017.pdf?file=1&title=National%20Retail%20Security%20Survey%202017.

6. Financial Crimes Enforcement Network (FinCEN), "Feasibility of a Cross-Border Electronic Funds Transfer Reporting System Under the Bank Secrecy Act: Appendix D – Fundamentals of the Funds Transfer Process," US Department of the Treasury, October 2006, https://www.fincen.gov/sites/default/files/shared/Appendix_D.pdf.

7. "Card Fraud Worldwide," *The Nilson Report*, PDF, October 2016, https://nilsonreport.com/upload/content_promo/The_Nilson_Report_10-17-2016.pdf.

8. Walmart, 2016 Annual Report, April 30, 2018, http://s2.q4cdn.com/056532643/files/doc_financials/2016/annual/2016-Annual-Report-PDF.pdf; The Kroger Company, 2016 Annual Report, April 30, 2018, http://ir.kroger.com/Cache/1500099541.PDF?O=PDF&T=&Y=&D=&FID=1500099541&iid=4004136; Costco Wholesale, Annual Report 2016, April 30, 2018, http://phx.corporate-ir.net/External.File?item=UGFyZW50SUQ9NjU1NTg1fENoaWxkSUQ9MzYxNDM2fFR5cGU9MQ==&t=1; The Home Depot, 2016 Annual Report, April 30, 2018, http://ir.homedepot.com/~/media/Files/H/HomeDepot-IR/reports-and-presentations/annual-reports/annual-report-2016.pdf; CVS Health, 2016 Annual Report, April 30, 2018, http://investors.cvshealth.com/~/media/Files/C/CVS-IR-v3/reports/annual-report-2016.pdf; Walgreens Boots Alliance, Annual Report 2016, April 30, 2018, http://files.shareholder.com/downloads/WAG/6215787588x0x920659/858BCE46-131D-4764-8410-1F35998DD1F8/278444_Final_BMK.pdf; Amazon.com, 2016 Annual Report, April 30, 2018, http://phx.corporate-ir.net/External.File?item=UGFyZW50SUQ9NjY2MjA0fENoaWxkSUQ9Mzc0MDUxfFR5cGU9MQ==&t=1; Target, 2016 Annual Report, April 30, 2018, https://corporate.target.com/_media/TargetCorp/annualreports/2016/pdfs/Target-2016-Annual-Report.pdf; Lowe's Companies, 2016 Annual Report, April 30, 2018, http://phx.corporate-ir.net/External.File?item=UGFyZW50SUQ9NjY3MDUzfENoaWxkSUQ9Mzc0ODQ3fFR5cGU9MQ==&t=1; Nasdaq, "Albertsons Companies, Inc. IPO Financials & Filings," April 30, 2018, https://www.nasdaq.com/markets/ipos/company/albertsons-companies-inc-970028-78908?tab=financials; Apple Inc., 2016 Annual Report, April 30, 2018, http://files.shareholder.com/downloads/AAPL/6219007589x0x913905/66363059-7FB6-4710-B4A5-7ABFA14CF5E6/10-K_2016_9.24.2016_-_as_filed.pdf; Seven & I Holdings & Co., Integrated Report 2016, April 30, 2018, https://www.7andi.com/dbps_data/_template_/_user_/_SITE_/localhost/_res/en/ir/library/ar/pdf/2016_10.pdf; Ahold Delhaize, Annual Report 2016, April 30, 2018, https://www.aholddelhaize.com/media/4045/ahold-delhaize_annual-report-2016_interactive.pdf; Wesfarmers, 2016 Annual Report, April 30, 2018, https://www.wesfarmers.com.au/docs/default-source/reports/2016-annual-report.pdf?sfvrsn=4; Woolworths Group, Annual Report 2016, April 30, 2018, https://wow2016ar.qreports.com.au/xresources/pdf/wow16ar-financial-report.pdf; Deborah Weinswig, "European Grocery Discounters: Small Stores—Big Threats?," Fung Business Intelligence Centre, PDF, November 2015, https://www.fbicgroup.com/sites/default/files/European%20

Grocery%20Discounters%20Report%20by%20FBIC%20 Global%20Retail%20Tech.pdf; Carrefour, 2016 Annual Activity and Responsibile Commitment Report, April 30, 2018, http://www.ecobook.eu/ecobook/Carrefour/2016/view/RA-EN.html; "Market Study: Food retail in Germany – market structure data 2016," bulwiengesa, PDF, June 21, 2017, https://www.tlg.eu/fileadmin/user_upload/Publikationen-en/pdf/2017_06_23_Food_retail_in_Germany_-_Market_structure_data_2016_EN.pdf; Tesco, Annual Report 2016, April 30, 2018, https://www.tescoplc.com/media/264194/annual-report-2016.pdf; Metro Group, Annual Report 2015/2016, April 30, 2018, http://reports.metrogroup.de/2015-2016/annual-report/servicepages/downloads/files/entire_metrogroup_ar16.pdf; Aeon Co., Ltd., Financial Results for the Fiscal Year ended February 28, 2017, April 30, 2018, http://v4.eir-parts.net/v4Contents/View.aspx?template=ir_material_for_fiscal_ym&sid=35790&code=8267; Auchan Holding, 2016 Annual Financial Report, April 30, 2018, https://www.auchan-holding.com/uploads/files/modules/results/1504611432_59ae8c68d5036.pdf; D&B Hoovers, "EDEKA ZENTRALE AG & Co. KG," April 30, 2018, http://www.hoovers.com/company-information/cs/company-profile.edeka_zentrale_ag__co_kg.109606d824e6baca.html; Groupe Casino, 2016 Registration Document, April 30, 2018, https://www.groupe-casino.fr/en/wp-content/uploads/sites/2/2014/01/Document-de-reference-EN.pdf.

9. Walmart et al.

10. Jonnelle Marte, "Google, Square thwarted by banks' 1970s tech," *MarketWatch*, November 5, 2013, https://www.marketwatch.com/story/banks-rely-on-1970s-tech-to-move-money-2013-10-21.

11. Carmen Chai, "Contactless 'tap-and-go' cards finally enter US market," *CreditCards.com*, November 15, 2017, https://www.creditcards.com/credit-card-news/contactless-tap-and-go-cards-us-market.php.

12. Chris Corum, "Apple to enable iPhone NFC tag reading in new models, still restricts card emulation," *SecureIDNews*, June 9, 2017, https://www.secureidnews.com/news-item/apple-to-enable-iphone-nfc-tag-reading-in-new-models-still-restricts-card-emulation/.

13. FinCEN.

14. FinCEN.

15. "Global Cards - 2016," *The Nilson Report*, https://nilsonreport.com/research_featured_article.php.

16. Zennon Kapron and Michelle Meertens, "Social Networks, e-Commerce Platforms, and the Growth of Digital Payment Ecosystems in China: What It Means for Other Countries," Better Than Cash Alliance, PDF, April 19, 2017, https://www.betterthancash.org/tools-research/case-studies/social-networks-ecommerce-platforms-and-the-growth-of-digital-payment-ecosystems-in-china.

17. FinCEN.

18. "SWIFT FIN Traffic & Figures," SWIFT, April 23, 2018, https://www.swift.com/about-us/swift-fin-traffic-figures.

19. "ACH Volume Grows to More Than 25 Billion Payments and $43 Trillion in Value in 2016," NACHA, April 12, 2017. https://www.nacha.org/news/ach-volume-grows-more-25-billion-payments-and-43-trillion-value-2016.

20. NACHA, "Same Day ACH: Moving Payments Faster," *Same Day ACH Resource Center*, April 9, 2016, https://web.nacha.org/resource/same-day-ach/same-day-ach-moving-payments-faster.

21. FinCEN.

22. Capgemini and BNP Paribas, "World Payments Report 2017," PDF, October 9, 2017, https://www.capgemini.com/en-us/wp-content/uploads/sites/2/2017/10/world-payments-report-2017_year-end_final_web-002.pdf.

23. Kim Zetter, "That Insane, $81M Bangladesh Bank Heist? Here's What We Know," *WIRED*, May 17, 2016, https://www.wired.com/2016/05/insane-81m-bangladesh-bank-heist-heres-know/.

24. Sriram Iyer and Anwesha Ganguly, "Everything you need to know about the $1.8 billion PNB-Nirav Modi fraud," *Quartz India*, February 16, 2018, https://qz.com/1208266/the-1-8-billion-punjab-national-bank-nirav-modi-fraud-explained/.

25. Chief General Manager, Jose J. Kattoor, "Press Release," Reserve Bank of India, February 20, 2018, https://rbi.org.in/scripts/bs_pressreleasedisplay.aspx?prid=43181.

26. SBI Holdings, Inc., "Financial Results," PDF, July 27, 2017, http://www.sbigroup.co.jp/english/investors/disclosure/presentation/pdf/170727presentations.pdf.

27. McKinsey & Company, "Global Payments 2016: Strong Fundamentals Despite Uncertain Times," PDF, September 2016, https://www.mckinsey.com/~/media/McKinsey/Industries/Financial%20Services/Our%20Insights/A%20mixed%202015%20for%20the%20global%20payments%20industry/Global-Payments-2016.ashx.

28. Ripple, "Solution Overview," PDF, April 30, 2018, https://ripple.com/files/ripple_solutions_guide.pdf.

29. Jeff John Roberts, "Western Union Is Testing Ripple and XRP for Money Transfers," *Fortune*, February 14, 2018, http://fortune.com/2018/02/14/ripple-xrp-western-union-money-transfers/.

30. "The transaction volume for UnionPay reached 14.95 trillion US dollars in 2017," UnionPay International, February 6, 2018, http://www.unionpayintl.com/en/mediaCenter/newsCenter/companyNews/3533.shtml.

31. "Global Cards - 2016."

32. "Annual Report 2017," VISA, PDF, January 30, 2018, https://s1.q4cdn.com/050606653/files/doc_financials/annual/2017/Visa-2017-Annual-Report.pdf.

33. "Global Cards - 2016."

34. "Merchant Customer Exchange," *Wikipedia*, https://en.wikipedia.org/wiki/Merchant_Customer_Exchange.

35. Jeff Bukhari, "That Chip on Your Credit Card Isn't Stopping Fraud After All," *Fortune*, February 1, 2017, http://fortune.com/2017/02/01/credit-card-chips-fraud/.

36. Al Pascual, Kyle Marchini, and Sarah Miller, "2018 Identity Fraud: Fraud Enters a New Era of Complexity," Javelin Strategy & Research, PDF, February 6, 2018, https://www.javelinstrategy.com/coverage-area/2018-identity-fraud-fraud-enters-new-era-complexity.

37. U.S. Payments Forum, "Card-Not-Present Fraud around the World," PDF, March 2017, https://www.uspaymentsforum.org/wp-content/uploads/2017/03/CNP-Fraud-Around-the-World-WP-FINAL-Mar-2017.pdf.

38. "Card Fraud Losses Reach $22.80 Billion," *The Nilson Report*, PDF, October 2017, https://nilsonreport.com/upload/content_promo/The_Nilson_Report_Issue_1118.pdf.

39. Ben Knieff, "2016 Global Consumer Card Fraud: Where Card Fraud is Coming From," Aite Group and ACI Worldwide, PDF, July 2016, https://www.aciworldwide.com/-/media/files/collateral/trends/2016-global-consumer-card-fraud-where-card-fraud-is-coming-from.pdf.

40. Identity Theft Resource Center and CyberScout, “2017 Annual Data Breach Year-End Review,” PDF, https://www.idtheftcenter.org/images/breach/2017Breaches/2017AnnualDataBreachYearEndReview.pdf.

41. Tara Siegel Bernard et al. “Equifax Says Cyberattack May Have Affected 143 Million in the U.S.,” *New York Times* online, September 7, 2017, https://www.nytimes.com/2017/09/07/business/equifax-cyberattack.html.

42. Miles Parks, “Target Offers $10 Million Settlement in Data Breach Lawsuit,” *NPR* online, March 19, 2015, https://www.npr.org/sections/thetwo-way/2015/03/19/394039055/target-offers-10-million-settlement-in-data-breach-lawsuit.

43. Nicole Perlroth, “All 3 Billion Yahoo Accounts Were Affected by 2013 Attack,” *New York Times* online, October 3, 2017, https://www.nytimes.com/2017/10/03/technology/yahoo-hack-3-billion-users.html.

44. “From Tickets to Tastings: American Express Introduces the Anniversary Collection to Celebrate 25 Years of the Membership Rewards® Program,” American Express Company, June 30, 2016, http://about.americanexpress.com/news/pr/2016/amex-celebrates-25-years-membership-rewards.aspx.

45. Kapron and Meertens.

46. Kapron and Meertens.

47. Mary Meeker, “Internet Trends 2017,” Kleiner Perkins Caufield & Byers, PDF, May 31, 2017, http://www.kpcb.com/file/2017-internet-trends-report.

48. Chief Executive Officer, Dan Schulman, “PayPal’s Fourth-Quarter and Full-Year 2017 Results,” PayPal, January 31, 2018, https://www.paypal.com/stories/us/paypals-fourth-quarter-and-full-year-2017-results.

49. “PayPal Q1-18 Investor Update,” PayPal, PDF, April 25, 2018, https://investor.paypal-corp.com/common/download/download.cfm?companyid=AMDA-4BS3R8&fileid=978256&filekey=00F974E6-C92A-44BE-82AE-3DF86E429149&filename=Investor_Update_First_Quarter_2018.pdf.

50. Pratik Bhakta, “Paytm transactions reportedly grew four-fold to $20 billion in February,” *The Economic Times*, March 20, 2018, https://economictimes.indiatimes.com/small-biz/startups/newsbuzz/paytm-transactions-reportedly-grew-four-fold-to-20-billion-in-february/articleshow/63375003.cms.

51. Pascual, Marchini, Miller.

The Harvest Finance Launch 🚜

Harvest automatically farms the highest yields in DeFi.

Harvest Finance Follow
Aug 29, 2020 · 4 min read

MAN AND FOOD
NO.1 EVOLUTION OF FARM POWER
Yoke 4000 B.C.
Agriculture began about 5500 B.C.
Horse collar 1000 A.D.
INDUSTRIAL REVOLUTION 1800
Steam farm power 1870
Gasoline farm power 1889
Kerosene farm power 1927 model
Heat Engines … basis for all modern farm power
Chemistry … pneumatic rubber tires made tractors more mobile sources of farm power
Metallurgy the design lightweigh
🚜 Harvest
Diesel, L.P. gas, kerosene or gasoline engines power the modern rubber-tired tractor
Diesel farm power was first introduced in crawler tractors. Gas turbine power is now being tested

Through untold ages, man labored alone in the soil. About 5000 B.C. man learned that animals could provide the main effort in working the soil. For the next 6800 years, however, human and animal muscles remained his only sources of power.

Following these centuries of slow progress, a rapid series of events known as the Industrial Revolution occurred. This era began about 1800 and its first impact on agriculture was felt with the advent of the iron plow. Farm mechanization began by the late 1870's and it is from this date that we trace *Farm Power*.

Within one lifetime, there has been more progress in agriculture than during all the preceding centuries.

🥖 Bread For The People 👩‍🌾👩‍🌾

📈 In the last few months, yield farming has become an unstoppable force. Humble farmers from all over the world have been tilling their crops to help feed themselves. Some get ample harvest, but many don’t!

Farmers are a diverse bunch, they have varying degrees of expertise and experience, and farming can prove to be very cumbersome for those farmers who lack access to skills, knowledge, tools and information.

The history of agriculture has been marked by technological advancements that allowed human populations to scale by maximizing the available yield through better tools and crop selection.

As you can see in the image above, we evolved from using primitive tools like the yoke to advanced machines like the tractor, which allowed humans to maximize yield and scale our population to billions. 🌾

With that, we present **Harvest** 🚜, a tool that helps farmers of all shapes and sizes get automatic exposure to the highest yield available across select decentralized finance protocols.

We hope this will make yield farming more accessible and help create a sustainable community-governed farming cooperative that only has one goal in mind: **#BreadForThePeople.** 🥖👩‍🌾👩‍🌾

📚 Protocol Design

- Harvest automatically farms the highest yielding assets and distributes the profits among the people. 🌿
- The harvesting strategies are flexible and future-proof. A majority of the past and upcoming assets can be farmed through Harvest. New crops with standard implementation can be farmed for you as they see the light of the world. Non-standard crops will be farmed as soon as respective strategies get developed.
- Harvest's clean and consistent design allows outside developers to easily add to it and receive rewards for their efforts. There is no time to waste while weeds are growing. 🌱
- The governance cannot drain staked assets or farmed crops from the farming strategies. Your beans will always be safe. 🧺

📈 Protocol Incentives

- In addition to the yields from harvesting, the protocol provides incentives to its users for making deposits. Users of Harvest receive $FARM 🚜.
- Protocol profits are distributed to the holders of $FARM which aligns incentives for Harvest users to govern and hold a stake in its continuous success.

👩‍🌾 Token Distribution

- Total $FARM supply: 5,000,000 FARM distributed over 4 years
- Circulating supply at launch: 0 FARM
- FARM is bootstrapped and has no VCs or investors
- Emissions happen as rewards are farmed:
- 70% for liquidity providers from incentive pools
- 10% rewards to operational treasury
- 20% rewards to team for building Harvest
- 👩‍🌾 FARM holders receive the 5% fee from Harvest operations.

💸 Reward Distribution

Through our automated yield farming, users will receive rewards depending on which pool they enter, which are automatically harvested into the base pool asset. In addition to the yields and rewards from harvesting, the protocol provides incentives to its users for making deposits. Users of Harvest receive $FARM 🚜. Through adding new pools and strategies, we will be able to keep expanding the list of reward assets.

Protocol Emissions for Bootstrapping Period:
Week 1: 57569.1
Week 2: 51676.2
Week 3: 41250.3
Week 4: 30824.4
Total for first 4 weeks: 181,320 (3.63% of supply)
After week 4: constant emission of 23555 FARM per week for 4 years

🏎 How You Can Participate

Harvest will launch in the next few days and will be available to all yield farmers who would like to use the protocol and participate in our cooperative.

The countdown clock has begun on https://harvest.finance/

The deposit page will open on Monday Aug 31st, at 7pm UTC.

FARM rewards for incentive pools begin on Tuesday Sep 1st, at 7pm UTC. Be sure to stake before that for maximum FARM yields 🚜.

#BreadForThePeople 🥖👩‍🌾👩‍🌾

📱 Discord and Twitter

- Join our Discord discussion at https://discord.gg/UZvqBjZ
- Don't miss an update by following us at https://twitter.com/harvest_finance

- Check out our Github https://github.com/harvest-finance/harvest

✌️Acknowledgements

Thanks to:

- The many people who helped out with this project, including providing comments and helpful suggestions for the early iterations of this product.
- Weeb, who has created one of the most useful public goods at https://yieldfarming.info/
- Andre Cronje, who provided inspiration in that a lone developer can build so much useful DeFi infrastructure so quickly. He’s a great example of what is possible in the world of DeFi where barriers to entry have been greatly reduced, and software, creativity, and value are so closely interlinked.
- The community for participating in this farming cooperative with us, thank you!

Tornado Cash Privacy Solution
Version 1.4

Alexey Pertsev, Roman Semenov, Roman Storm

December 17, 2019

1 Introduction

Tornado.Cash implements an Ethereum zero-knowledge privacy solution: a smart contract that accepts transactions in Ether (in future also in ERC-20 tokens) so that the amount can be later withdrawn with no reference to the original transaction.

2 Protocol description

The protocol has the following functionality:

- Insert/deposit money to the smart contract. This can be done in a single transaction with a fixed amount (denoted by N) of Ether. The N-ETH note is called a *coin*.
- Remove/withdraw money from the smart contract can be done in 2 ways:
  - The N ETH is withdrawn through a Relayer with f Ether sent as a fee to the Relayer address t and $(N - f)$ to the designated recipient. The value f and t is chosen by the sender. In this case the withdraw transaction is initiated by the Relayer and it pays the Gas fee that is supposed to be covered by f.
  - The N ETH is withdrawn to the designated recipient, the transaction is initiated by the recipient. The recipient should have enough ETH to pay Gas fee for the transaction. In that case fee f is considered to be equal to 0.

2.1 Setup

Let $\mathbb{B} = \{0, 1\}$. Let e be the pairing operation used in SNARK proofs, which is defined over groups of prime order q.

Let $H_1 : \mathbb{B}^* \to \mathbb{Z}_p$ be a Pedersen hash function defined in [Ped]. Let $H_2 : (\mathbb{Z}_p, \mathbb{Z}_p) \to \mathbb{Z}_p$ be the MiMC hash function [AGR+16] defined as a MiMC permutation in the Feistel mode in a sponge mode of operation[1].

Let $\mathcal{T}$ be a Merkle tree of height 20, where each non-leaf node hashes its 2 children with H_2. It is initialized with all leafs being 0 values. Later the zero values are gradually replaced with other values from $\mathbb{Z}_p$. Let $O(\mathcal{T}, l)$ be the Merkle opening for leaf with index l (value of sister nodes on the way from leaf l to the root, denoted by R) in tree $\mathcal{T}$.

Let us call $k \in \mathbb{B}^{248}$ a *nullifier* and $r \in \mathbb{B}^{248}$ a *randomness*. Let us denote an Ethereum address of the coin recipient by A.

Let $\mathcal{S}[R, h, A, f, t]$ be the following statement of knowledge with public values R, h, A, f, t:

$$\begin{aligned}\mathcal{S}[R, h, A, f, t] = \{&\text{I KNOW } k, r \in \mathbb{B}^{248}, l \in \mathbb{B}^{16}, O \in Z_p^{16} \text{ SUCH THAT } h = H_1(k) \\ &\text{AND } O \text{ is the opening of } H_2(k||r) \text{ at position } l \text{ to } R\}\end{aligned} \quad (1)$$

[1] https://github.com/iden3/circomlib/blob/master/src/mimcsponge_gencontract.js

where A and f are included into the context of the statement. Here h is called *nullifier hash* and $||$ is concatenation of bitstrings.

Let $\mathcal{D} = (d_p, d_v)$ be the ZK-SNARK [Gro16] proving-verifying key pair for $\mathcal{S}$ created using some trusted setup procedure. Let $\text{Prove}(d_p, \mathcal{T}, k, r, l, A, f, t) \to P$ be the proof constructor using d_p and $\text{Verify}(d_v, P, R, h, A, f, t)$ be the proof verifier.

Let $\mathcal{C}$ be the smart contract that has the following functionality:

- It stores the last $n = 100$ root values in the history array. For the latest Merkle tree $\mathcal{T}$ it also stores the values of nodes on the path from the last added leaf to the root that are necessary to compute the next root.
- It accepts payments for N ETH with data $C \in \mathbb{Z}_p$. The value C is added to the Merkle tree, the path from the last added value and the latest root is recalculated. The previous root is added to the history array.
- It verifies the alleged proof P against the submitted public values (R, h, A, f, t). If verification succeeds, the contract releases $(N - f)$ ETH to address A and fee f ETH to the Relayer address t.
- It verifies that the coin has not been withdrawn before by checking that the nullifier hash from the proof has not appeared before and if so, adds it to the list of nullifier hashes.

2.2 Deposit

To deposit a coin, a user proceeds as follows:

1. Generate two random numbers $k, r \in \mathbb{B}^{248}$ and computes $C = H_1(k||r)$
2. Send Ethereum transaction with N ETH to contract $\mathcal{C}$ with data C interpreted as an unsigned 256-bit integer. If the tree is not full, the contract accepts the transaction and adds C to the tree as a new non-zero leaf.

2.3 Withdrawal

To withdraw a coin (k, r) with position l in the tree a user proceeds as follows:

1. Select a recipient address A and fee value $f \leq N$;
2. Select a root R among the stored ones in the contract and compute opening $O(l)$ that ends with R.
3. Compute nullifier hash $h = H_1(k)$.
4. Compute proof P by calling Prove on d_p.
5. Perform the withdrawal in one of the following ways:
   - Send an Ethereum transaction to contract $\mathcal{C}$ supplying R, h, A, f, t, P in transaction data.
   - Send a request to Relayer supplying transaction data R, h, A, f, t, P. The Relayer is then supposed to make a transaction to contract $\mathcal{C}$ with supplied data.

The contract verifies the proof and uniqueness of the nullifier hash. In the successful case it sends $(N - f)$ to A and f to the Relayer t and adds h to the list of nullifier hashes.

3 Implementation

The cryptographic functions for off-chain use are implemented in the circomlib library[2]. The Solidity implementation of Merkle tree, deposit, and withdraw logic is by the authors[3]. The Solidity implementation of MiMC is by iden3[4]. The SNARK keypair and the Solidity verifier code are generated by the authors using SnarkJS. The other protocol logic (e.g., Ethereum transaction composition, SNARK proof construction calls) is by the authors[5].

[2] https://github.com/iden3/circomlib/tree/master/circuits
[3] https://github.com/tornadocash/tornado-core/tree/master/contracts
[4] https://github.com/iden3/circomlib/blob/master/src/mimcsponge_gencontract.js
[5] https://github.com/tornadocash/tornado-core/blob/master/cli.js

4 Security claims

Tornado claims the following security properties:

- Only coins deposited into the contract can be withdrawn;
- No coin can be withdrawn twice;
- Any coin can be withdrawn once if its parameters (k, r) are known unless a coin with the same k has been already deposited and withdrawn.
- If k or r is unknown, a coin can not be withdrawn. If k is unknown to the attacker, he can not prevent the one who knows (k, r) from withdrawing the coin (this includes all cases of front-running a transaction).
- The proof is binding: one can not use the same proof with a different nullifier hash, another recipient address, or a new fee amount.
- The cryptographic primitives used by Tornado have at least 126-bit security (except for the BN254 curve where the discrete logarithm problem has something like 100-bit security), and the security does not degrade because of their composition.
- For each withdrawal every deposit since the last moment when the contract has zero Ether till the formation of the root in the proof can be a potential coin, though some coins are more likely to be withdrawn depending on the user behaviour.

References

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, et al. “MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity”. In: *ASIACRYPT (1)*. Vol. 10031. Lecture Notes in Computer Science. 2016, pp. 191–219 (cit. on p. 1).

[Gro16] Jens Groth. “On the Size of Pairing-Based Non-interactive Arguments”. In: *EUROCRYPT 2016*. Vol. 9666. LNCS. Springer, 2016, pp. 305–326 (cit. on p. 2).

[Ped] *Iden3: Pedersen Hash*. `https://iden3-docs.readthedocs.io/en/latest/iden3_repos/research/publications/zkproof-standards-workshop-2/pedersen-hash/pedersen.html`. 2019 (cit. on p. 1).

NEXUS MUTUAL

A peer-to-peer discretionary mutual on the Ethereum blockchain.

HUGH KARP, REINIS MELBARDIS

ABSTRACT

The insurance industry has developed over time from a community-based model to an adversarial one where large institutions dominate. It is also inefficient in many areas leading to large frictional costs being borne by customers. Blockchain technology allows individuals to efficiently transact directly with each other and therefore enables the core insurance entity to be replaced. Nexus Mutual uses blockchain technology to bring the mutual ethos back to insurance by creating aligned incentives through smart contract code on the Ethereum blockchain.

BACKGROUND

Before insurance companies existed, communities would group together themselves. They would pool resources to protect individual members from risks they all faced.[1] If an unfortunate event occurred the senior members of the community would decide whether to provide assistance or not. All funds raised were used to benefit the members of the community.

In developed nations we have largely moved away from this community approach primarily due to the underlying economics of insurance. Insurance economics are driven by diversification. The more individual risks that are pooled together the less capital is required to be confident all claims can be met.[2] Scale benefits are significant and community models don't have the means to access them easily.

Moving away from the community model brought other challenges, in particular the issue of agency. An insurer is looking after customers money and then promising it will pay when a claim arises. As a result, the insurer is becoming an agent of the customer and history has proven this model doesn't work without heavy oversight from government institutions and complex legal frameworks. These frameworks are necessary primarily due to the lack of trust between customers and the institution and boil down to two main points:[3]

1. AGENCY - Insurers decide on how customers money is handled. Including how it is invested, which insurance risks it will back and when it gets paid out to shareholders. They also have an implied option where there is potentially unlimited upside but if the insurance company goes bust it is customers that suffer. Interests are not directly aligned.
2. TRANSPARENCY - A customer finds it extremely difficult to assess how safe a particular insurer is. There is a clear information asymmetry issue.

In developed nations both of these issues are dealt with primarily via law and prudential regulation – a complex combination of standards defining minimum capital levels, governance processes, reviews and regular financial reporting. Regulation in this way is largely effective, barring a handful of high

[1] https://en.wikipedia.org/wiki/Mutual_insurance

[2] https://en.wikipedia.org/wiki/Law_of_large_numbers

[3] http://fsi.gov.au/publications/

profile exceptions[4], but brings additional costs and reduced flexibility.

Even with this burden the institutional model has provided significant benefits to customers via reduced premiums and deeper pockets. The underlying diversification benefits have more than outweighed the regulatory burden. But there is still substantial unnecessary cost in the system. Roughly 35%[5] of insurance premiums are lost due to frictional costs in the system. Only 65% of premiums are returned to customers via claims, the rest is lost in distribution, operational expenses (including regulatory), capital costs and profit.

Blockchain technology and smart contracts can strip out not only the administrative inefficiencies but a large portion of the governance and regulatory related costs. They can do this by providing trust in a different, much more cost-effective way. Trust is moved from institutions and regulations to transparent code. Of the 35% of frictional costs we believe blockchain technology can cut out approximately 18%[6] due to administrative savings and reduced governance and regulatory costs, effectively halving the frictional costs in the system.

Additionally, through the use of membership tokens, blockchain technology can bring back the original goals of the mutual where all contributions are entirely for the benefit of members. Aligned incentives will foster a community spirit rather the existing adversarial and unbalanced relationship between individual and large institution.

Blockchain technology allows a peer-to-peer insurance mutual to be recreated in a cost effective and scalable way. It allows the cooperative ethos to be regained while preserving the benefits of diversification.

Solution Overview

The following components are necessary for a peer-to-peer risk sharing mutual:

1. Membership Tracking – A way to track individual members, including their proportional ownership.
2. Claims Assessment Methodology – A way for claims to be approved or declined.
3. Capital Model – To define how much capital is required to back the risks at any point in time.
4. Funding – Ability to attract capital to back the risks and reward that capital appropriately for the risks taken. Initially and on an ongoing basis.
5. Investment Returns – Insurers hold customers money until a claim event occurs. During this time they tend to invest these funds, usually quite conservatively, to earn additional return.
6. Product – A viable product to sell, including underwriting rules and other acceptance criteria.
7. Pricing – A method for determining the fair risk charge for the risk cover and a way for it to adjust over time.
8. Distribution – Tools and incentives to attract new members to the mutual.
9. Identity – An identity module will be required as part of the sign-up process to conform with legal and regulatory requirements.
10. Governance – A way to upgrade, enhance and fine-tune the code in line with the wishes of the membership base,

[4]https://en.wikipedia.org/wiki/List_of_corporate_collapses_and_scandals

[5]http://www.mckinsey.com/industries/financial-services/our-insights/what-drives-insurance-operating-costs

http://www.guycarp.com/content/dam/guycarp/en/documents/dynamic-content/Insurance_Risk_Benchmarks_Research_Annual_Statistical_Review.pdf

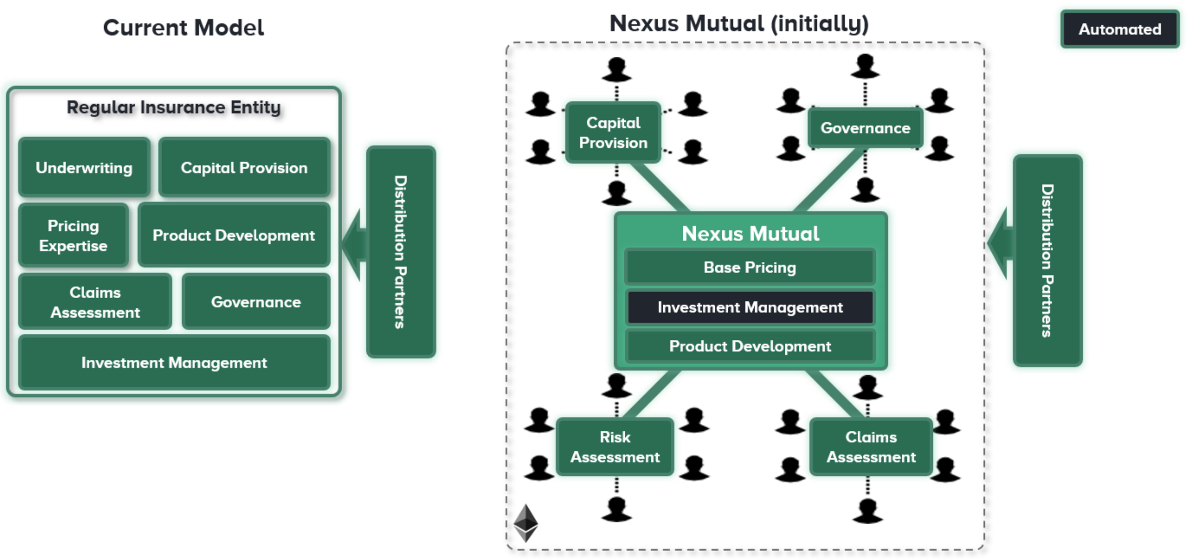
[6] See Appendix A

as well as the ability to interact with the non-blockchain world.

11. TRANSPARENCY – Real time reporting of capital position and risk exposures.
12. LEGAL FRAMEWORK – A safe legal and regulatory environment to operate within.

The next sections of the paper will describe each of these components in turn, followed by additional comments on the competitive strategy.

A visual overview of the general structure, is shown below:

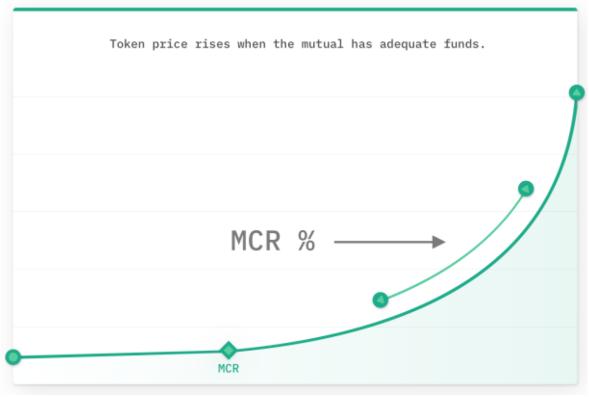


MEMBERSHIP

A simple ERC-20 compatible token will be created to serve as the key internal incentive mechanism to bind the mutual together.

A continuous token model will be used so that tokens can be purchased at any time but at a variable price. This contrasts to more common ICO type approaches where there is a fixed purchase period with set price change points, followed by a speculation-driven market on exchanges.

The token price will vary based on 1) funding level of the Capital Pool and 2) the minimum amount of capital required to support existing covers (which provides a link to business growth):



Note: Diagram illustrates funding level only. Price also varies with the amount of capital required to support existing cover.

$$TP = A + \frac{MCR_{ETH}}{C} \cdot MCR\%^4$$

TP = *Token Price in Ether*

MCR_{ETH} = *The minimum amount of capital required to support existing covers, Minimum Capital Requirement, in Ether. The MCR is calibrated to a 99.5% solvency level.*

MCR% = *Ratio of Capital Pool funds to the Minimum Capital Requirement.*

A and C = *Fixed constants, to be calibrated based on the prevailing Ether price before launch.*

Tokens can only be created in the following ways:

1. INITIAL TOKENS – Some tokens will be set aside for founders and early contributors when the contract is deployed.
2. PURCHASED VIA THE TOKEN PRICE MODEL – Anyone, at any point, can purchase tokens via the token price model. When funding is required (ie low MCR%) the price will be lower to encourage funds to be placed. Conversely the token price increases when funds are more plentiful. Price also increases based on the business growth (represented by growth in the MCR) which places a natural throttle on token issuance. The token model ensures a balance is reached between adequate compensation for the risks taken by early participants and allowing future members to join at any time.
3. CLAIMS ASSESSMENT REWARDS – Additional member tokens are allocated as an incentive to perform claims assessment. This will be limited to a fixed percentage of the cost of cover.
4. RISK ASSESSMENT REWARDS – Additional member tokens are allocated as an incentive for participating in risk assessment.
5. GOVERNANCE – Additional member tokens are allocated as an incentive for participating in governance.

While the supply of member tokens is not fixed all methods of generating new member tokens require a specific contribution to the mutual. Contributions are made as either funds or services (claims assessment, risk assessment or voting in governance).

Membership tokens can be used in the following ways:

1. PURCHASING COVER – Member tokens can be used ("burned") to purchase cover. In this case the token value is determined

by the continuous token model. 90% of the tokens used are burned, with the remaining 10% locked for the cover period plus 35 days, as they are required to submit a claim.

2. Claims Assessment Stake – To participate in claims assessment and earn the resulting income, member tokens must be staked.
3. Risk Assessment Stake – To participate in assessing risks and earning commissions a stake is required.
4. Redemption - If the Capital Pool has sufficient funds redemptions of member tokens in exchange for Ether is permitted.

   The following restrictions will apply:

   1. Capital Pool needs to be above the MCR (MCR% > 100%).

   2. Redemptions are capped per transaction.

   3. The Capital Pool must have enough liquidity in Ether.

   4. Sell price will be 2.5% below the prevalent buy price.

Only members of the mutual will be able to own tokens. As such, tokens cannot be transferred to any Ethereum address that has not been designated as a member.

Claims Assessment

There are two main approaches to claims assessment using blockchain technology. Firstly, using an oracle which is either a trusted off-chain information provider (eg to trigger parametric insurance events) or secondly, crowd-sourcing information and assessing claims using voting mechanics (eg a prediction market).

Under a discretionary mutual model there is a legal requirement that a group or sub-group of members decide on how funds are distributed. This immediately focusses efforts on the crowd-source approach but there are other arguments that limit the usefulness of parametric trigger-based cover:

1. Basis Risk[7] - This can lead to poor customer outcomes especially when customers have suffered a loss but the trigger has not technically been met.
2. Oracle Failure - Back-up claims process mechanisms will be required if the oracle were to fail.
3. Limited Product Set – Product development requires a reliable data oracle to exist. The data must also be sufficiently granular to construct a meaningful consumer product. IoT devices are expected to bring many more potential data oracles in the future but are currently not widespread or reliable enough.

Returning to the crowd-source model, there needs to be an incentive for people to report and a strong disincentive to prevent fraudulent reporting. This is somewhat difficult to achieve in an insurance context because there is a clear incentive to defraud the pool by 1) purchasing cover for a low percentage of the cover amount, 2) using a substantial portion of the cover amount to pay-off claims assessors and then 3) pocketing the difference.

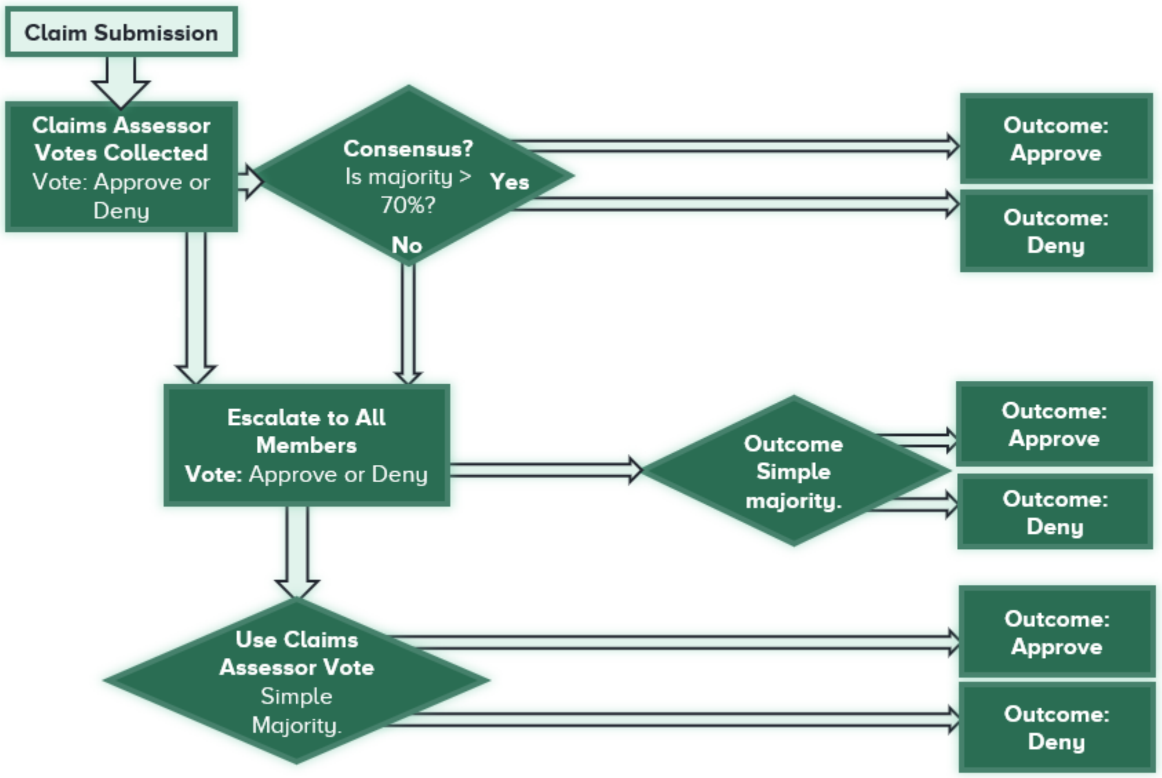
A solution to this issue is to require claims assessors to have a significant stake in the success of the overall pool and a high disincentive to act dishonestly. This can be achieved by requiring a stake be posted in the form of membership tokens. The stake is deposited for a specified period of time and provided claims are assessed honestly it is returned. If the Advisory Board deems a claims assessor to be acting dishonestly it has the power to burn the staked member tokens.

[7] https://www.questia.com/library/journal/1P3-1252828171/understanding-basis-risk-in-insurance-contracts

In addition, the following other incentive structures will be put in place:

- Voting with the consensus outcome entitles claims assessors to a share of the fee pool. Fees will be paid as additional member tokens and valued at a fixed percentage of the cost of cover.
- Voting against the consensus outcome results in locking of the bond for a longer period. Assessment is often challenging and automatically burning high values of member tokens for genuine differences of opinion needs to be avoided.
- Voting power must add up to greater than 5x the cover amount, where voting power is determined by the number of staked member tokens used to vote.
- No consensus results in a reduced fee pool for claims assessors and the claim is then escalated to all members for a vote.
- Member tokens contributing to claims assessment voting become "inactive" and cannot contribute to another claims assessment for 12 hours. This prevents posting a sufficiently high stake, submitting many fraudulent claims of total value well above the staked amount and then approving them all. The Advisory Board has time to step in and burn tokens before too many fraudulent claims are approved. In this case the members would benefit overall as the accretive benefit from the burned member tokens would outweigh the fraudulent claims cost.
- Calibrations of the incentive mechanisms need to be refined in testing.

Designing incentive structures resilient to game theoretic attacks is very challenging. The approach described has a basic incentive structure at its core and then overlays timing windows and human intervention to prevent more extreme scenarios.


Claim Submission
Claims Assessor Votes Collected Vote: Approve or Deny
Consensus? Is majority > 70%?
Yes
No
Outcome: Approve
Outcome: Deny
Escalate to All Members Vote: Approve or Deny
Outcome Simple majority.
Outcome: Approve
Outcome: Deny
Use Claims Assessor Vote Simple Majority.
Outcome: Approve
Outcome: Deny

Capital Model

The capital model determines the minimum capital the fund needs to hold. The funding rules in the next section then reference the Minimum Capital Requirement (MCR) and determine actions such as the token price and redemption restrictions.

The capital model will borrow heavily from EIOPA's Solvency II[8] methodology which is calibrated to withstand events in a year with a 99.5% probability, or, in other words, a 1-in-200 year event. This is consistent with many other regulatory standards of nations such as Australia[9], Bermuda, Japan, Mexico and Singapore who either have specific targets of 99.5% or are on the way to gaining "equivalence" with the SII regime.

An alternative approach is to 100% collateralise the insurance contracts, essentially holding the full sum assured value at all times. In combination with the immutability of the blockchain this would give the consumer an extremely high level of security. This comes at the cost of severely reduced capital efficiency and the ability to raise funds at an appropriate price. As a simple example, assume we have 10,000 (n) identical policies each with a chance of claim of 1% (p) for a sum assured of $100 (v). Assuming independence the 99.5% Minimum Capital Requirement (MCR) is given by:

Mean = $\mu = p \cdot n = 100$

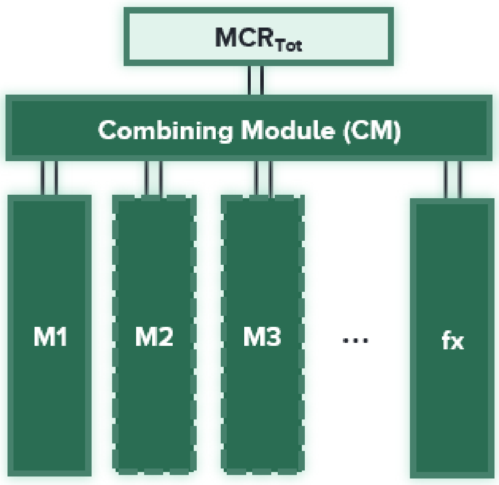
Std Dev = $\sigma = \sqrt{n \cdot p \cdot (1-p)} = 9.9499$

MCR = $v \cdot (\mu + 2.58 \cdot \sigma) = \$12{,}567$

Total Exposure = $n \cdot v = \$1{,}000{,}000$

In this example, we expect 1% of the total exposure to be paid out in claims, but with 10,000 contracts we only need 1.26% of the total exposure to be confident the fund will be solvent in 199 out of 200 scenarios. This diversification benefit needs to be leveraged otherwise we cannot compete with existing institutions.

The capital model is structured in multiple modules, where each module represents a product and currency pair. In addition, there is a currency module (fx) to account for currency risk. The modules are then combined at a total level to get the MCR. In its simplest form, with one product and one currency there are three modules, M1, fx and CM.



The base calculation currency is Ether as the pool will be Ether dominated to start with. The MCR of each individual module is calculated in its currency (ie ETH or DAI[10]) and then converted to Ether in the combining module.

Focussing on module one to begin with, and assuming the product has a fixed sum assured MCR_{M1} is defined as follows:

$MCR_{M1} = \sqrt{\sum_{i,j} Corr(i,j) \cdot Exp(i) \cdot Exp(j)}$

Where;

[8] https://eiopa.europa.eu/regulation-supervision/insurance/solvency-ii

[9] http://www.apra.gov.au/Policy/Documents/Regulation-Impact-Statement-LAGIC.pdf

http://www.aon.com/attachments/reinsurance/052011_ab_latin_america_solvency_regulation_paper_051911.pdf

https://www.munichre.com/site/corporate/get/documents_E-2113795143/mr/assetpool.shared/Documents/5_Touch/_Publications/302-08131_en.pdf

[10] https://makerdao.com/whitepaper/DaiDec17WP.pdf

Corr(i,j) is the correlation matrix of the individual pricing risk cells;

$$\text{Corr(i,j)} = \begin{bmatrix} 1 & \cdots & corr(j,i) \\ \vdots & \ddots & \vdots \\ corr(i,j) & \cdots & 1 \end{bmatrix}$$

And Exp(i) = Total probability-weighted exposure (or cover amount) in pricing risk cell i.

The correlation matrix may be very simple if independence between cells can be assumed in which case MCR_{M1} reduces to:

$$MCR_{M1}= \sqrt{\sum_i Exp(i)}$$

It is possible that each product module may have a different formulaic logic to get to an assumed 99.5% confidence capital requirement. In particular, this would be required with indemnity-based products rather than fixed cover amount values.

The next step is the currency module (fx) which takes the MCR's of each module in a particular currency (k), compares that to the value actually held in the pool, V_j, and applies a currency shock of 50%, both up and down, and then chooses the maximum value. The sum of all these becomes MCR_{fx}:

$$MCR_{fx} = \Sigma_k \mid (\Sigma_k MCR_i - V_k) / 50\% \mid \cdot fx_{k \text{ to } \Xi}$$

Where $fx_{k \text{ to } \Xi}$ is the exchange rate to Ether.

The combining module then takes a similar approach to the MCR_{M1} calculation, treating each module as its own pricing risk cell and assuming a correlation between different modules:

$$MCR_{Tot}=\sqrt{\sum_{l,m} Corr(l,m) \cdot MCR(l) \cdot MCR(m)}$$

subject to a minimum value.

Where, Corr(l,m) is the correlation matrix of the modules:

$$\text{Corr(l,m)} = \begin{bmatrix} 1 & \cdots & corr(l,m) \\ \vdots & \ddots & \vdots \\ corr(l,m) & \cdots & 1 \end{bmatrix}$$

A minimum MCR value will be set when the pool launches and the MCR value can never drop below this.

The total MCR will need to be calculated regularly, probably at least once per day, as it is needed as a reference item for funding triggers. Operationally this will work as follows:

- Calculation will need to be performed off-chain, due to gas requirements, with the result being notarised on-chain.
- The capital model code will be open-source and all inputs will be available on-chain (either directly or via oracles for currency exchange rates) or as part of the model itself.
- Correct running of the model will be verified on-chain.
- Updates to the model or input parameters will be handled via the governance process.
- There will be a specified block number on which calculations are made. This locks the data inputs to the calculation model and gives enough time for the model to be run off-chain.
- To begin with it is likely the MCR will be run in a trusted manner off-chain due to technical limitations. In the future trust minimising options for complex computation will be investigated further with a strong intention to remove this reliance.

FUNDING

The funding levels are all effectively governed by the continuous token model described in the membership section. The total Capital Pool value is V, which is calculated as the sum of all the asset values converted into Ether.

When the fund is first launched no covers can be purchased until an MCR% of 100% is achieved (which will be once the Capital Pool is equal to the Minimum Capital Requirement). Once that happens the fund goes live and the token model interacts with the capital model to increase or decrease the token price as required.

Another aspect of funding is the multi-currency pool of funds. As member fees and claim payments will be constantly flowing in and out of the pool, rules are required (both trigger limits and targets) to ensure the right level of funds are held in each currency. Also, as the capital model punishes mismatches in fund pools vs MCRs by currency modules (via greater MCR_{Tot}) a decision on allocation is required. Targets and trigger limits will be set, which can be updated via the governance process as necessary.

Additionally, some trust-less way of converting fiat-crypto tokens to Ether is required to balance the pool. As per the investment returns section, this will be handled via the Uniswap[11].

More broadly, there is an implicit assumption throughout the paper regarding the availability of a fiat-based crypto token for all currencies the mutual wishes to trade in. At present no widely adopted solutions to this exist, though many companies and organisations have publicly stated they are developing solutions and MakerDAO has recently gone live with DAI (a USD stable-coin). Initially, Nexus Mutual will use Ether and DAI ($USD) as its initial currencies and wait for further solutions to develop and become more widely adopted.

Investment Returns

Investment returns are an often under-appreciated aspect to insurance as it allows the insurance entity to earn returns on the reserves it holds. This is a key component to insurers' profitability and therefore must be replicated in some fashion if Nexus Mutual is able to compete with existing insurance entities longer term.

As Nexus Mutual will hold all funds on-chain, it will restrict itself to assets of ETH and ERC20 tokens only. At present this asset universe is quite small but it is expected to grow substantially over time.

The investment process will be entirely automated using the Uniswap protocol to initiate trades. A buy and hold investment strategy will be defined and trades will rebalance the pool as required. There will also be trading triggers to deal with liquidity needs arising from claim payments. The assets chosen will need to change over time, with the changes initiated and approved via the governance module.

Such an approach means basic investment management can be entirely automated and conducted in a trust-less way.

Ideally, the assets would generate a positive return over time with very high probability, akin to the portfolio composition of traditional insurers which tend to be dominated by corporate and government debt instruments[12]. In the Ethereum world, we see the current most appropriate candidates for generating a return on ETH as:

- locking up ETH to generate interest in the proposed Proof of Stake system,
- investing in financial instruments based on collateralised lending[13]
- acting as a guarantor in state channel and payment channel networks.

Unfortunately, we are still some time away from Ethereum moving to a Proof of Stake system. With insufficient scale and liquidity currently available in the various ETH-based lending markets, becoming a payment channel guarantor is more likely to be viable

[11] https://github.com/Uniswap/uniswap-info

[12]http://www.oecd.org/investment/Evolution-insurer-strategies-long-term-investing.pdf

[13] https://dharma.io/

in the short term, but the technology still needs to mature and be adopted more widely by other blockchain applications. The current lack of investment options is not considered a major issue in the short term given the expected short policy durations of the initial product. It is therefore likely that Nexus Mutual will initially launch without any investment assets, only holding currency assets closely matched to the liabilities of the mutual.

An alternative approach would be to invest a portion of the funds into a basket of ERC20 tokens, in the hope that they gain in value relative to ETH. We do not see any reason to believe that such investments exist and, if they do, that we would be able to pick out such a basket from the outset. However, such investments could be made via the member governance process.

PRODUCT

Initially the mutual will be launched with only one product, Smart Contract Cover with a fixed cover amount. The product will cover "unintended code usage" where someone, not necessarily the cover purchaser, has suffered a financial loss on the smart contract. As an example, the cover would pay out on the DAO hack, and the two Parity multi-sig wallet issues. It is not intended to pay-out on loss/misuse/phishing of private keys as this cannot be verified.

This product is seen to have a very good market fit for the early adopters of the platform. Security of smart contracts is a well-publicised issue within the Ethereum community with many technical efforts being led to improve the situation. Longer term, the intention is to expand the product range into more regular insurance products and become an alternative risk carrier for the insurance industry.

The initial product has been chosen for several reasons:

- Claims assessment can be done entirely remotely using publicly available data from block explorers.
- A fixed cover amount means claims assessment is a simple "yes" or "no" rather than requiring an assessment of how much damage has been caused.
- The product pricing can be largely automated allowing covers to be issued without any mandatory manual underwriting.
- It is not necessary to confirm the member has an insurable interest in the specific contract.
- The product is new to market with no alternatives existing. Many developers are very worried about deploying code to main-net, as even with many security audits and thorough testing you can never be completely sure bugs don't exist.
- The likely short-term nature of the covers is a good fit given the (lack of) on-chain investment options available.

Numerous future products can be developed such as indemnity-based cover, life cover, auto cover etc. Many of them will require some form of initial underwriting process and much more complex claims assessment procedures. The goal is to initially build a product with a clear consumer need in our target audience before expanding into regular product lines.

PRICING & CAPACITY LIMITS

Given the lack of historic data on smart contract hacks, related information on code security will be used to assist pricing. Additionally, it is expected that most new contracts will start off with a very high (or even uninsurable) cost which is then reduced over time as the code is more battle-tested. However, by itself this is not of any material benefit to code developers as they will often want cover immediately.

Therefore, we are introducing the concept of decentralised risk assessment, which involves knowledgeable experts (think smart contract security auditors) staking value in the form of member tokens against specific risks to reduce the price of cover.

If there is an early claim then part or all of the stake will be lost. In return, the risk assessor will earn commission in the form of tokens for cover sold on that particular address.

In this way, we are combining a standard pricing algorithm with decentralised risk assessment to develop a complete pricing framework.

Another important risk mitigation technique employed by the mutual involves capacity limits. A relatively simple approach will be taken whereby exposure to any single smart contract (or related and very similar contracts) will be limited to a fixed percentage of the pool value. This ensures that any one claim event does not put the solvency of the mutual at risk.

From an upgrade perspective, any member can propose a detailed one-off review of pricing at any time. This would re-set the base pricing with a new set of rates/algorithm. Alternatively, pricing can be provided off-chain via an API. This option is a likely first improvement step which is easier to implement and more flexible but introduces a level of trust in the API.

Unlike traditional insurers, pricing will also be flexible enough for cover periods in daily increments, with a formula used to determine rates for specific, non-yearly cover periods.

Distribution

Distribution will initially focus on the relatively small group of cryptocurrency enthusiasts, entirely within the cryptocurrency sphere. This will enable any teething issues to be identified before building out more products and attempting significant scaling by offering the product to a mainstream audience. There is ample opportunity in the short to medium term to provide meaningful growth with the initial product, in particular:

- Well-funded projects looking to deploy code could purchase cover in case something goes wrong. This would help minimise reputational damage and provide funds to compensate users if necessary.
- Individuals looking to interact with smart contracts may want extra confidence before exposing funds. Very few individuals have the capability to assess code quality by themselves. This is especially important when large values are involved.
- Projects launching an ICO looking to provide confidence to prospective funders may want to pre-purchase cover for their ICO contract code.
- Multi-sig wallet contracts could be insured. While not addressing private key management issues this gives greater confidence funds won't just disappear. This could form part of a more comprehensive custody solution designed by 3rd parties.

Distribution in the short term will come primarily via community engagement and promotion within the cryptocurrency community driven from within the project.

Longer term, when the product range is expanded to more typical products the main challenges to wider distribution are:

- Accessing Crypto Tokens – As future products require purchasing fiat-crypto tokens the development of consumer wallet tools and processes is needed to achieve any meaningful scale. Approaches whereby distribution partners handle the crypto aspects and allow consumers to conduct business entirely outside the crypto sphere will be the primary focus.

- FIXED SUM ASSURED – Most consumers are accustomed to indemnity-based products where claims paid cover losses actually incurred.
- DISTRIBUTION PARTNERS – Many insurance policies are sold through brokers, so enabling an attractive financial distribution model will be key to attracting larger volumes. Distribution tools and marketing material will need to be developed.

In summary, the longer-term vision is not for products to be mass marketed to consumers directly, but rather as a B2B2C platform that distribution partners can integrate with via blockchain's inherent open API architecture. This is similar in nature to the concepts behind existing insurance distribution and the latest trends in open-banking in the UK.

Therefore, a key aspect to the long-term success of the mutual are the distribution partners. The smart contract platform is designed to be as open as possible and therefore quite flexible for distributors to interact as they see fit (subject to any compliance obligations).

IDENTITY

It will be necessary to identify all members of the mutual. This is because each member becomes a guarantor of the company and is required by company law in the UK to be identified.

There will be a simple identity process where KYC is conducted that links an Ethereum address to the customer, noting that all identifying information is not held on-chain. This will be a one-time process when signing up as a member.

From then on, the Ethereum address will be linked to the member. This serves a dual purpose of legal compliance and providing some level of Sybil attack prevention, noting that the system is designed to be Sybil resistant anyway.

GOVERNANCE

Ideally all potential actions can be defined by the code but reality is much more complex and fall-back options are required in several circumstances. As such an Advisory Board will be set-up to facilitate decisions requiring interaction with the non-blockchain world as well as govern some of the more extreme scenarios. Importantly, the Advisory Board has no custodial rights over the fund pool and cannot release funds to any particular person, with each Board member liable to be replaced at any time via the member voting process.

The Advisory Board will operate under two core principles:

1. SUSTAINABILITY – Protect existing members by ensuring the overall fund is sustainable; and
2. GROWTH - Enable sustainable premium and member growth.

At the start, it will contain several individuals who are all members of the mutual and contain a mix of expertise within insurance, mutual governance and blockchain development.

Advisory Board members will have the following broad authorities, which will be specified in more detail:

1. Facilitate and implement the wishes of the membership base, particularly where the code doesn't specifically allow automatic implementation.
2. Punish bad actors within the Claims Assessment process.
3. Meet all the legal and regulatory requirements of Nexus Mutual Ltd.
4. Implement emergency pause functionality if required.
5. White-list and vote on proposals where required.

A detailed list of authorities will lay out what Advisory Board members can agree on by themselves and what proposals need to go to members for a final decision.

All proposals put to a member vote must contain a defined list of the possible voting outcomes (eg Yes/No) as well as the Advisory Board recommendation and vote result. Members are then given a specified timeframe to vote on the proposal. The majority outcome prevails unless a specified quorum is not met – then the vote proceeds as per the Advisory Board recommendation.

Individual members can develop proposals for the Advisory Board who will have some discretion whether to "white-list" the proposal or not. The aim is to "white-list" all reasonable proposals.

Any individual member may propose that they join the Advisory Board. This type of proposal is automatically put to a full member vote without proceeding through the "white-listing" process. This ensures the members ultimately maintain full control of the mutual as any Advisory Board member can be replaced without interference from the Advisory Board.

Transparency

A key requirement for operating a well-run mutual entity is providing members, potential members and other interested parties with accurate information regarding the financial health of the mutual. Blockchain technology lends itself quite naturally to transparency due to the public ledger. As such, a website interface will be developed which reports on key metrics in real-time. These will include information such as:

- Capitalisation ratio (MCR%).
- Exposure by pricing cell, and groupings.
- History of capital metrics and token price.
- Number of total member tokens outstanding split by locked vs transferrable.
- Details on claims assessment results, with summary statistics.

In combination this information will provide an accurate real-time financial position of the mutual. Compared to a regular insurer's financial reporting, which generally takes 3 months (at best) to determine a quarterly result, blockchain can provide orders of magnitude improvement in both timeliness and transparency.

Legal Framework

Nexus Mutual has been set-up as a company limited by guarantee in the UK and will operate under a discretionary mutual structure. Members of the mutual will have a legal right to proportional ownership of the mutual and will also be responsible for providing the guarantee.

The guarantee will be set at £1 per member. This means if the mutual was ever to run out of money, each member is liable for a further £1 only.

A discretionary mutual is not a provider of insurance, it is a legal structure that enables members to trade with each other under the banner of one legal personality. Therefore, it is not required to conform with all the insurance regulatory and legal requirements. In addition, products are not subject to Insurance Premium Tax (IPT) in the UK with any distributions or surplus being taxed in the hands of members. The mutual will pay tax on any trade outside of the mutual, for example VAT on services and corporate tax on investment income.

A discretionary mutual based in the UK can legally trade in the UK but cover can be provided anywhere in the world. As such, global cover is available as long as;

1. Members are able to legally become a member of the UK company, and;

2. Local laws and regulations of the members jurisdiction are adhered to.

Practically, this means Nexus Mutual will be able to provide cover in most countries with some being restricted for various local legal reasons, such as securities laws, insurance regulation and tax.

As a real world legal entity, the mutual can interact directly with non-blockchain service providers as well as regulated insurance entities. The latter is particularly useful as excess-of-loss insurance coverage may be required for high exposures to facilitate faster growth

Nexus Mutual will adhere to the principles in the Association of Financial Mutuals (UK industry trade body) code of conduct and will investigate the process of becoming a full member.

All of the above views are formed based off informed research and discussion with business and legal experts. While many aspects have also been verified through formal legal advice there still remains uncertainty with how products and platforms like Nexus Mutual interact with the legal system, especially as many aspects still require guidance from various regulators. As such, when joining, any members of the mutual agree that they will withdraw their membership should it be required for legal or regulatory reasons that would endanger the ongoing operation of the mutual. Nexus Mutual fully intends to comply with all regulation.

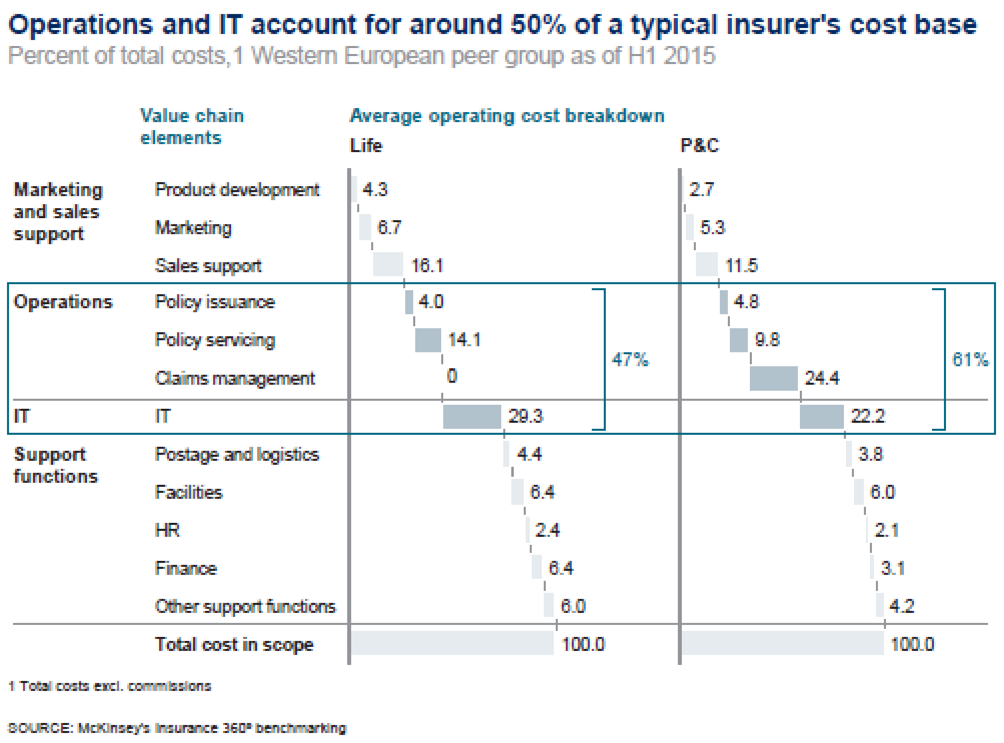
COMPETITIVE STRATEGY

A key challenge in open source business is retaining a competitive advantage when anybody can copy your entire code base, decrease margins slightly and poach all your customers. To remain relevant the business must establish meaningful barriers to potential competition. In open-sourced blockchain systems this is largely achieved through the network effect where a community gathers around a certain technology, becomes bought into it (usually financially as well as emotionally and philosophically) and continuously improves it to remain relevant. The following barriers and frictional costs are designed to keep Nexus Mutual relevant to current members and continually attract new ones:

- RISK ASSESSOR NETWORK – Establishing a meaningful network of risk assessors (smart contract auditors to begin with) and providing them adequate incentives to participate.
- SIZE OF CAPITAL POOL – The faster scale can be achieved the larger the Capital Pool can grow and the greater the diversification benefits. This ensures efficient capital usage, lower prices and provides more resilience to claims shocks. Additionally, the greater the pool value the higher the barrier to replicate.
- CONTINUAL DEVELOPMENT – A continued focus on improvement of the product. Releasing new products and providing easy to use infrastructure surrounding the core blockchain code will heighten the barrier to replicate. This will be increasingly driven by all members of the mutual over time.
- MEMBER TOKENS – All customers are members and have a vested interest in the success of the mutual through token ownership. If members shifted to another provider their current holdings would drop in value. Membership tokens therefore provide an indirect incentive to remain with the mutual and an additional barrier to competitors.

Whilst all of these barriers have the potential to be overcome the goal is to gain network effects and scale benefits that will prevent copy-paste competitors taking significant market share.

APPENDIX A – COST REDUCTION ENABLED BY BLOCKCHAIN TECHNOLOGY



[14]

Focussing on the P&C column (Property and Casualty, i.e. short-term non-biometric insurance more akin to the initial offering of Nexus Mutual), the costs in the above diagram account for roughly 25% of premium, representing most of the ~35% of premium that gets lost in frictional costs[15]. The most notable cost excluded from the above is commission.

MARKETING AND SALES SUPPORT – These costs will largely remain as is for mainstream products. There are likely to be some small savings in sales support costs due to efficiency in the underlying systems but there won't be any material savings overall.

OPERATIONS AND IT – The major area where large cost savings can be realised. The only material costs that affect the proposed mutual will be gas costs, rewards for decentralised claim assessment and smart contract upgrades. We estimate these costs are reduced by 90%, as policy issuance and servicing are entirely automated, claims management is simplified and crowdsourced and systems normally required by insurers are made vastly more efficient by availability of the distributed ledger.

SUPPORT FUNCTIONS – Large cost savings will materialise across a number of sub-functions primarily because the number of people employed will be dramatically reduced. Only the Advisory Board is required at the start, with potential for some support functions if the marketing and sales support teams have grown large enough. We assume 90% of these costs can be avoided.

[14] http://www.mckinsey.com/industries/financial-services/our-insights/what-drives-insurance-operating-costs

[15] Typically, claims costs account for about 65% of insurance premium income (e.g http://www.guycarp.com/content/dam/guycarp/en/documents/dynamic-content/Insurance_Risk_Benchmarks_Research_Annual_Statistical_Review.pdf), with expenses making up the rest up to the point where typically most of the premium income gets spent (e.g. https://www.verisk.com/siteassets/media/downloads/insuranceresultsreport2016q4.pdf).

Therefore, combining the above estimates, we expect to reduce the non-commission frictional costs by approximately 72% compared to a traditional insurance company. Converting it back to a percentage of premium income, this equates to a further 18% of premiums accruing in the mutual for the benefit of the members.

Note that the above discusses a comparison to established insurance companies assuming comparable products and sales channels applying to Nexus Mutual. Initially, the cost base is likely to be reduced further due to the niche nature of the product resulting in pre-incurred product development costs and a fully digital marketing approach aimed at the blockchain community.

creamY Launches in the next 24 hours with CREAM Incentives

C.R.E.A.M. Follow
Sep 23, 2020 · 2 min read

We are excited to launch creamY, our capital efficient, dynamic AMM, beginning with the stablecoin market. Join us as we deploy this new AMM with some novel innovations — dynamic pools, consolidated liquidity, single-sided liquidity adds, all in a yielding, stable LP token we call cyUSD that also works well as a stablecoin.

Background

A few days ago we announced creamY, our capital efficient, dynamic AMM here. Andre Cronje also did a technical write up here. We are launching the creamY stablecoin market first and will launch the BTC and ETH markets soon thereafter. We also shortened the LP token prefix to cy- from cry- to reduce name collisions.

CREAM Rewards

We will start with a strong, 7-day incentive program, and optimize the incentives quickly. These two pools will be available at launch tomorrow:

Pool 1 — Stake cyUSD and share a portion of the 500 CREAM/day pool.

Pool 2 — Add liquidity to the 95/5 cyUSD/CREAM pool, stake your CRPT token and share a portion of the 1,500 CREAM/day pool.

There will be no locks with these opening pools. We will add more, longer duration pools as adoption of creamY plays out.

A Word of Caution

Even though this code has been reviewed thoroughly by several credible developers, this code has not yet been through formal audit nor production testing. We are pushing this code through audits now, and will provide updates as we progress. Please do not put in more money than you can afford to lose.

C.R.E.A.M. DAO
Crypto Rules Everything Around Me, C.R.E.A.M.

Join us on Discord, follow us on Twitter, or visit us at cream.finance.

Ethereum Bitcoin Blockchain Defi Finance

dYdX: A Standard for Decentralized Margin Trading and Derivatives

Antonio Juliano

September 25, 2017 *[Updated August 6, 2018]*

Abstract

We present a set of protocols that allow several types of financial products to be created, issued, and traded for any pair of underlying ERC20 tokens. Our approach uses off-chain order books with on-chain settlement to allow creation of efficient markets. All described protocols are fair and trustless, creating truly open markets that are not governed by a central authority. The protocols are extensible by anyone, requiring no special permissions to be used with other smart contracts.

Contents

1 Introduction

The rise of blockchains has enabled anyone to own and transfer assets across an open network without needing to trust any external parties. Unlike existing financial architecture, blockchains are freely and equally available worldwide. This has led to a large and rapidly increasing number of digital assets existing on the blockchain. Many centralized and decentralized platforms designed to facilitate the efficient exchange of these assets already exist, and more are in development. Such platforms allow investors to take long positions in various assets. However, it is currently very difficult or impossible to take more complex financial positions.

dYdX allows creation of entirely new asset classes which derive their value from underlying blockchain-based assets. Financial products such as derivatives and margin trades allow investors to achieve superior risk management with their portfolios, and open up new avenues for speculation. They also increase market efficiency for the underlying asset by aiding in price discovery and allowing individuals to express more complex opinions on price and volatility. dYdX provides advantages over traditional financial products by eliminating the need for a regulated central clearing house, providing global and equal access, and allowing users full control of their funds at all times.

The size of the derivatives market on existing financial infrastructure far outstrips the market size of any other type of financial asset. It is roughly estimated to be over $1.2 quadrillion[1], or more than 10 times the total world GDP. We believe that as decentralized platforms mature and start to offer significant advantages over traditional financial systems, an ever increasing number of traditional assets will start to be listed on the blockchain.

dYdX will offer a number of decentralized protocols implementing various types of crypto-asset financial products. These protocols are comprised of open source Ethereum Smart Contracts and standards.

[1] Investopedia. *How big is the derivatives market?*.
http://www.investopedia.com/ask/answers/052715/how-big-derivatives-market.asp

2 Existing Work

There are few existing decentralized protocols that support derivatives or margin trading and none that have any significant usage. Centralized exchanges also fail to offer adequate financial products on decentralized assets. Consequently, it's very difficult to take short or more complex financial positions on the bulk of today's decentralized assets.

In order for a decentralized derivatives or margin trading protocol to operate, there needs to be a way to trustlessly exchange assets, as well as determine the price at which assets will be exchanged. A decentralized exchange protocol is one that facilitates the trustless exchange of one token for another at prices dictated by the market. dYdX can work with any standard Ethereum-based decentralized exchange. Initially, dYdX will use the 0x protocol[2] to enable token exchange at rates supplied by users of the protocol.

Several types of decentralized exchanges have been proposed: on-chain order books, automated market makers, state channels, and a hybrid off-chain order book approach. The 0x whitepaper offers an in-depth discussion of the tradeoffs between these models[3]. We chose to base dYdX on the hybrid approach pioneered by 0x, as we believe it allows creation of the most efficient markets. This allows market makers to sign and transmit orders on an off-blockchain platform, with the blockchain only used for settlement.

One previous attempt at decentralized derivatives, Velocity[4], proposed using an oracle based approach to feed the exchange rates of asset pairs to a smart contract responsible for operation of options contracts. The contract would then use this price information to create and exercise options. Using such an oracle based approach has several significant drawbacks. The limitations on frequency, latency, and cost of price updates due to the nature of blockchains makes it impossible to create markets as efficient as those built on traditional centralized exchanges. Using an oracle also adds a great deal of centralization to any protocol, as some central parties have full control over setting the price. Worse, if those central parties were also trading on the protocol, they would have a huge economic incentive to manipulate prices in their favor.

dYdX protocols allow trade of financial products at any price agreed upon by two parties. This means that there is no need for the contracts to be aware of the market price. Traders provide orders of their choosing, which are then used to execute the exchange. It is in the economic interest of traders to choose orders with the best prices. This best price is dictated by the market, and no orders with better prices will exist.

[2] Will Warren, Amir Bandeali. *0x White Paper*. https://0xproject.com/pdfs/0x_white_paper.pdf
[3] Will Warren, Amir Bandeali. *0x White Paper*. "Existing Work". https://0xproject.com/pdfs/0x_white_paper.pdf
[4] Shayan Eskandari, Jeremy Clark, Vignesh Sundaresan, Moe Adham. *On the feasibility of decentralized derivatives markets*. https://users.encs.concordia.ca/~clark/papers/2017_wtsc.pdf

3 Protocols

dYdX consists of a number of protocols specifying the operation and execution of different types of financial products. We plan to prioritize the development of the most popular and widely used types. Below we outline our implementation of protocols for options and margin trades. We plan to develop protocols for additional types of financial products in the future.

3.1 Margin Trading Protocol

3.1.1 Description

In a margin trade, a trader borrows an asset and immediately trades it for another asset. The asset must be repaid to the lender, usually along with interest, at a later date. Margin trading includes both short sells and leveraged longs.

In a short sell an investor borrows an asset and sells it for the quote currency. The investor makes money if the price of the asset decreases, since rebuying the asset to repay the lender costs less than the original sell-price.. The investor loses money if the price of the asset increases, since rebuying the asset to repay the lender costs more than the original sell-price. The lender makes money from the interest paid by the trader.

In a leveraged long an investor borrows the quote currency and uses it to buy an asset. The investor makes money if the price of the asset increases, and loses money if it decreases. Gains or losses from the position are equal to the change in price of the underlying asset multiplied by the leverage ratio, which is the ratio of the sum of the borrowed amount plus the amount paid by the trader to the amount paid by the trader.

3.1.2 Use Cases

Short sells are used to enable investors to profit from an asset which decreases in price. Short sells can be used for both speculation and hedging. Investors can use a short sell for speculation when they believe the price of an asset will go down. Short sells can be used to hedge existing positions by shorting a correlated asset.

Leveraged longs are used to multiply gains when an asset increases in price. Leveraged longs can be used for speculation, as they allow traders to achieve larger gains with less capital. Investors can use leveraged longs for more efficient capital allocation, as less capital is required to achieve the same results for each investment.

Lending assets for margin positions can provide the lenders with interest from the loan.

3.1.3 Overview

The dYdX Margin Trading protocol uses one main Ethereum Smart Contract to facilitate decentralized margin trading of ERC20 tokens. Lenders can offer loans for margin trades by signing a message containing information about the loan such as the amount, tokens involved, and interest rate. These loan offers can be transmitted and listed on off-blockchain platforms.

A trader opens a margin position by sending a transaction to the dYdX margin smart contract containing a loan offer, a buy order for the borrowed token, and the amount to borrow. Upon receiving this transaction, the smart contract transfers the margin deposit from the trader to itself, and then uses an external decentralized exchange such as 0x to sell the loaned token using the specified buy order. The smart contract holds onto the deposit and token resulting from the sale of the loaned token for the life of the position.

The position is closed when the trader sends a transaction to the smart contract containing a sell order offering to sell the amount of token owed to the lender for an amount less than or equal to the amount locked in the position. Upon receiving this transaction, the contract uses an external decentralized exchange to execute the trade between the order maker and itself. After, the contract sends the owed amount of the loaned token to the lender. The amount owed to the lender includes the interest fee. The trader is sent all of the leftover token, which is equal to $deposit + profit$. Note the profit could be negative if the price moved against the position.

The loan for a margin trade can also be called in by the lender when the price has moved against the position. Once the loan is called in, the trader has a specified amount of time to close the position. The trader can also allow other contracts to trustlessly and automatically close the position on their behalf using mechanisms such as a dutch auction.

The margin trading protocol can be used for both short selling and leveraged long trading by simply switching which token is borrowed (referred to as the *owed token*) with the one that is held in the position (referred to as the *held token*)[5]. The protocol allows the margin deposit to be paid in either *owed token* or *held token*. If the deposit is paid in *owed token* it is sold along with the *owed token* borrowed from the lender, so that only *held token* is held in the position. Similarly, the payout to the trader from closing can be in either *owed token* or *held token*. If the payout is in *owed token*, all *held token* in the position is sold for *owed token* and whatever is leftover after paying the lender is paid out to the trader.

When used for short selling, the trader will borrow *base token* from the lender which will be sold for *quote token*, and put up a margin deposit in *quote token*. Only *quote token* is held in the position. When the position is closed *base token* will be bought and paid back to the lender, and the trader will be paid out in *quote token*.

[5] The concept of *owed token* and *held token* should not be confused with *base token* and *quote token*. Depending on whether it is a short sell or leveraged long position, *base* or *quote token* can be the *owed* or *held token* in the position. This section aims to articulate the relatedness of the two positions from an implementation point of view.

When used to take a leveraged long position, the trader will borrow *quote token* from the lender, and put up a margin deposit in *quote token*. Both the *quote token* borrowed from the lender, as well as the *quote token* put up as margin deposit are then sold for *base token*. Only *base token* is held in the position. When the position is closed *all* of the *base token* is sold for *quote token*. *quote token* is paid back to the lender, and the trader is again paid out in *quote token*.

3.1.4 Implementation

3.1.4.1 Contracts

For margin trading, there are three contracts used: the *Margin* contract, the *Proxy* contract, and the *Vault* contract.

The *Proxy* is used to transfer user funds. Users set token allowances on the *Proxy* which authorizes it to transfer funds on their behalf.

The *Margin* contract offers functionality to enable margin trading. It contains all the business logic and public functions. It also contains the state where positions are stored. The *Margin* contract is designed so existing positions cannot be modified by any external party (see the governance section).

The *Vault* contract holds all the funds locked up in positions. It exposes a simple interface which the *Margin* contract is authorized to use.

3.1.4.2 Offering Message

The first ingredient to a margin trade is a lender who holds the *owed token*, and wants to lend it out for a given deposit and interest rate. The lender prepares and cryptographically signs a message with the following information:

Name	**Type**	**Description**
owedToken	address	Address of *owed token* - the token borrowed from and owed to the lender
heldToken	address	Address of *held token* - the token held in escrow by the position
payer	address	Address that supplies the funds for the loan. If this is different than *signer* it is assumed to be a smart contract and its consent is gotten through an interface
signer	address	Address that cryptographically signs the loan offering
owner	address	Address that will own the loan after it is taken. All payouts will

		go to this address
taker	address *(optional)*	If set, only this address will be able to take the loan
feeRecipient	address *(optional)*	Address to receive relayer fees associated with this offering
lenderFeeToken	address *(optional)*	Address of the token to charge the lender fee in
takerFeeToken	address *(optional)*	Address of the token to charge the taker fee in
maxAmount	uint256	The maximum amount of the loan offering. Denominated in units of *owed token*
minAmount	uint256	The minimum takeable amount of the loan offering. Denominated in units of *owed token*
minHeldToken	uint256	The minimum amount of *held token* locked in the position after the deposit and sell (based on *maxAmount*)
lenderFee	uint256 *(optional)*	Amount of *lenderFeeToken* to charge the lender (based on *maxAmount*)
takerFee	uint256 *(optional)*	Amount of *takerFeeToken* to charge the taker (based on *maxAmount*)
interestRate	uint32	The interest rate (continuously compounded, represented as annual nominal percentage with up-to 6 decimal places)
interestPeriod	uint32 *(optional)*	The interest rate update period. Interest fee will increase once per period
expirationTimestamp	uint32	The timestamp (in seconds since unix epoch) at which the offering expires
callTimeLimit	uint32	The minimum amount of time (in seconds) that the position must be closed after being margin-called by the lender
maxDuration	uint256	The maximum duration (in seconds) of the loan. Relative to when a position is opened

This message can then be broadcast off-blockchain between counterparties. It is a binding agreement to commit to the loan if a trader desires. The protocol is agnostic to the medium of exchange used to relay these signed messages. It is expected that these offers will be listed on centralized platforms referred to as relayers and will compete on interest rate and terms. Larger OTC trades can be agreed upon through traditional means, then made formally-binding using the protocol.

3.1.4.3 Buyer

The second ingredient to a margin trade is a buy order which can be filled as part of the margin trade.Like the loan offering, the buy order can be transmitted through any means. The buyer is in no way involved in the loan or margin trade. This order can be for any price, and must be selected by the trader. The only prerequisite is the order must be for at least as much *owed token* as the trader is selling as part of the margin trade. It is in the trader's economic interest to select the buy order with the best price.

dYdX allows any standard buy/sell decentralized exchange to be used. This is done by wrapping external decentralized exchange smart contracts in another contract that provides standard interface to *Margin*. The wrapping contract is known as an *ExchangeWrapper*. The *ExchangeWrapper* is specified by the trader for each margin trade and requires no special permissions. This means anyone can write, deploy, and use an *ExchangeWrapper* for any decentralized exchange. dYdX has implemented the first *ExchangeWrapper* which wraps the 0x Exchange Contract, and allows any 0x order to be used to open a dYdX position.

3.1.4.4 Position Opening

To open a position, a trader sends a transaction to the *Margin* smart contract containing:

- The signed loan offering
- The buy order offering to buy *owed token* for *held token*
- The address of the *ExchangeWrapper* to be used with the buy order
- The amount of *owed token* the trader wishes to borrow
- A boolean indicating whether the trader wishes to post margin deposit in *held token* or *owed token*
- The amount of token the trader wishes to put up as a deposit
- The address that will own the position after it is opened

When the contract receives the transaction the following happens:

1. The signature and inputs on the loan message are verified
2. *Margin* calls into *Proxy* to transfer the offered deposit in either *held token* or *owed token* from the trader to *Vault* (if depositing in *held token*) or to the *ExchangeWrapper* (if depositing in *owed token*)
3. *Margin* calls into *Proxy* to transfer the requested amount of the *owed token* from the lender to the *ExchangeWrapper*

4. *Margin* records that the requested amount of the loan has been used, and saves it in a mapping. This is used to keep track of the amount remaining in the loan offer and protect against replay attacks using the signed loan message[6]
5. *Margin* calls into the *ExchangeWrapper* to exchange the *owed token* for the amount of *held token* offered by the buy order. The buyer is the maker in this trade and the *ExchangeWrapper* is the taker. The exchange contract (e.g. the 0x Exchange Contract if using 0x) will verify the inputs and signature on the supplied buy order and execute the trade
6. *Margin* calls *Proxy* to transfer the *held token* received from the sell from the *ExchangeWrapper* to *Vault*. The *held token* remains locked in *Vault* for the duration of the position
7. The details of the position are stored in the contract, mapped by a unique public identifier for the position. This identifier is used by the trader and/or lender to interact with the position at a later date

All steps happen atomically, meaning that they all succeed or all fail together. At the end, the *Vault* contract ends up with an amount of *held token* for the position. If the margin deposit was put up in *held token*, this amount is equal to the deposit put up by the trader plus the *held token* resulting from the sale of the *owed token*. If the margin deposit was in *owed token*, the amount is equal to the *held token* resulting from the sale of both the borrowed *owed token* as well as the *owed token* put up as margin deposit. *Vault* holds onto these funds until the position is closed.

3.1.4.5 Closing

The trader can decide to close any portion of the position at any time by presenting the *Margin* contract with a sell order offering to sell greater than or equal to the amount of *owed token* owed to the lender (including interest fee) for an amount of *held token*. This sell order can be for any price such that there is enough *held token* in the position (prorated by the portion of the position being closed) to pay for it. However it is in the trader's economic interest to select an order with the lowest price.

When *Margin* receives this transaction, the following happens:

1. The total amount (in *owed token*) owed to the lender at this point in time is calculated using continuously compounded interest
2. *Margin* calls into the *ExchangeWrapper* to execute the trade of *held token* for the amount of *owed token* owed (if paying out in *held token*), or to trade all of the *held token* in the position for *owed token* (if paying out in *owed token*). After the trade, *Vault* holds the owed amount of *owed token* and an amount of either *held token* or *owed token* equal to $deposit + trader\ profit$ (profit could be negative) for the position
3. *Margin* calls into *Proxy* to transfer the owed amount of *owed token* from the *ExchangeWrapper* to the lender
4. *Margin* sends either *held token* or *owed token* equal to $deposit + profit$ to the trader

[6] Will Warren, Amir Bandeali. *0x White Paper*. "Fills & Partial Fills". https://0xproject.com/pdfs/0x_white_paper.pdf

5. *Margin* deletes the position from its storage if its value is now zero, or reduces its amount by the amount that was closed

At the end of the margin trade, the trader ends up with the $profit$ amount, denominated in either *held token* or *owed token*. The lender makes the amount of interest fee in *owed token*. The *Margin* / *Vault* contracts end up net neutral as desired.

3.1.4.6 Calling

The other way a margin trade can be settled is by the lender or another party authorized by the lender calling in the loan from the trader. It is done by the lender or authorized party sending the *Margin* contract a transaction indicating they are calling in the loan, along with an amount of *held token* that must be deposited into the position by the trader to cancel the margin call. After this transaction the trader has the amount of time originally specified in the loan (call time limit) to either pay back the loan, or put up additional *held token* as deposit. The trader uses the same process described above in the closing section to close the position. If the trader fails to close the position or put up the required additional deposit, the lender is entitled to the entire *held token* balance locked in the position.

It is in the lender's interest to call in the position when the price of *owed token* relative to *held token* rises to the point that the *held token* locked in the position is almost not enough to buy back the owed amount of *owed token*. This means the lender or authorized party needs to be watching the price and be ready to call in the position on an upward price movement. The authorized party would most likely be a relayer or service that watched the price and was always ready to programmatically call in loans on price movements. The approach of authorizing a trusted party is functionally equivalent to using a centralized oracle to determine when margin calls should occur, however is more efficient as gas does not need to be paid for constant price updates and the price can be effectively watched in real-time rather than once per block.

This approach also requires that the trader is always online and able to send a transaction to close the position before the call time limit, or risk forfeiting the entire position balance. To protect traders from always having to be online, traders can optionally opt-in to an external contract that can automatically and trustlessly close the position on their behalf.

The automatic closing contract works by running a dutch auction offering to buy back the amount of *owed token* owed to the lender for an amount of *held token* that starts at 0 and linearly increases to the total amount of *held token* in the position over the call time limit. Any excess *held token* is given to the trader. Anyone can bid on the auction, or use an existing decentralized exchange order to buy the *owed token* and keep the spread. The moment the auction price crosses the market price, there will exist an incentive for everyone to bid on the auction, causing the trader to get paid out at market price.

3.1.5 Risks

One risk for the trader is that the lender calls in the loan before the trader wishes to close the position even when enough deposit is posted. Current non-blockchain related financial systems use a reputation system to identify optimal lenders that will not call in the loan prematurely. Such a reputation system for dYdX could exist entirely separately to the base protocol, as traders would prefer loan offers from lenders with higher ratings and would price this into their decision on whether or not to take a loan. Another solution is to use an authorized party that is mutually trusted by both the trader and lender to margin call the position. In the future, decentralized price oracles could also be used.

The risk for the lender (besides the economic risk of holding the *owed token*) is that the price of the *owed token* relative to the *held token* rises so rapidly that the loan is not able to be margin-called before the amount of *held token* locked in the position is no longer enough to buy back the owed amount of *owed token*. In this case the lender would still receive the entire amount of *held token* locked in the position, but would have been better off just holding the *owed token*. This risk for the lender can be mitigated by setting a high enough deposit, low enough call in time, and by using an efficient margin-calling mechanism (likely through off-blockchain monitoring).

3.2 Options Protocol

3.2.1 Description

In an option, a holder of an asset sells the right to buy or sell that asset at a specified strike price and future date[7]. An option to buy an asset is referred to as a call, and an option to sell an asset is called a put. The seller of the option (the writer) collects a premium upon sale, but is also bound to buy or sell the asset at the agreed upon price and date if the holder of the option desires. A covered option indicates that the underlying asset is put up as collateral, so it is guaranteed to be able to be collected at a future date. The option can itself be traded on the open market. We describe an implementation of an American covered option, or one which can be exercised at any time before the expiration date.

3.2.2 Use Cases

Options enable numerous trading strategies that can be designed for speculation or risk management.

Options can be used to provide additional leverage in speculation. For example suppose the price of AAPL is $100, and an investor who has $1000 to invest believes it will go up. The investor could buy 10 shares at $100, and if the price rises to $110, selling would yield a $100 or 10% profit. Suppose instead that the investor had purchased call options with a $100 strike and $2 premium. The investor could afford 500 of these options with $1000. If the price again rose to $110, the investor could exercise the options to buy at $100, and then immediately sell at $110 for a $10 profit per option. Since the investor had paid $2 for each option, a profit of $8 per option would have been made. This means the investor's profit would have been $8 * 500 = $4,000 or a 400% return. This shows how with the same amount of capital investors can achieve much larger returns using options than by simply holding the asset.

Options can also be used to hedge or reduce risk in an investment. Imagine an investor is long 100 share of AAPL, which is again trading at $100. The investor could purchase a put option with $90 strike for a $2 premium. Such an option would ensure that for only a 2% fee, during the lifetime of the option the investor could not lose more than 10% on the investment.

Options also enable more advanced trading strategies such as straddles, strangles, collars, and many more. Among other things, such strategies can lock in a price, profit from volatility in any direction, or profit from price stability in an asset.

3.2.3 Overview

The dYdX option protocol uses one Ethereum Smart Contract per type of option. A type refers to a given set of input parameters including the *base token*, *quote token*, strike price, and expiration date. *base token* refers to the asset the option is for and *quote token* refers to the token in which the premium and strike

[7] Investopedia. *Options Basics: What are Options?*. http://www.investopedia.com/university/options/option.asp

price are denominated[8]. Each option contract is able to issue new options of its type at any time before the option expiration date. The contracts can act as either a put or a call option by simply switching the *base token and quote token* and inversing the strike price.

Writers of the option list offers for a specified lot size and premium on an off blockchain platform. Buyers can buy options from a writer by sending a transaction containing a write offer to the smart contract. After receiving such a transaction, the smart contract transfers the premium in *quote token* to the writer, and the offered amount of *base token* to itself. The buyer is issued options which can be transferred and traded as any other ERC20 token. The smart contract holds on to the *base token* until the option is either exercised or expired.

Any holder of the option can choose to exercise at any time before the expiration date. Upon exercise, the option holder pays $strike\ price \times (\#\ options)$ of *quote token* to the smart contract and is sent $\#\ options$ of *base token* from the smart contract. The *quote token* paid to the contract is distributed to the writer or writers of the option. After the option expires, all writers can withdraw *base token* from the smart contract corresponding to $\frac{Options\ Written}{Total\ options\ Written} \times (total\ tokens\ held)$.

3.2.4 Implementation

3.2.4.1 Contracts

We use three types of smart contracts to allow the issuance and functionality of options: the *Creator*, *Proxy*, and *CoveredOption* contracts.

The *Creator* is responsible for creating all *CoveredOption* contracts. Anyone can create a new type of *CoveredOption* by providing the the following specifications:

- The address of the ERC20 token the option is for (referred to as *base token*)
- The address of the ERC20 token the strike price and premium are to be paid in (referred to as *quote token*)
- The strike price (broken into two parts to form an exchange rate between *base token and quote token*)
- The expiration date

Creating a new type of *CoveredOption* only opens it up for sale, and does not issue any options. There can exist only one *CoveredOption* for each combination of input parameters.

The *Proxy* is responsible for transferring user tokens between accounts. Users use the ERC20 allowance functionality to authorize the *Proxy* to move their tokens. Each new *CoveredOption* is authorized to use the *Proxy* to transfer user funds when it is created by the *Creator*.

[8] Investopedia. *Base Currency*. http://www.investopedia.com/terms/b/basecurrency.asp

The *CoveredOption* contract represents a specific type of covered option. Each one implements the ERC20 interface to allow shares of the option to be traded and transferred after issuance. This means every option can be publicly traded on an exchange as any other ERC20 token.

3.2.4.2 Issuance

CoveredOption uses the exchange functionality of the 0x Protocol to facilitate issuance of new options. Options can be issued anytime before the expiration date of the option. In order to issue new options, the writer broadcasts a signed message in the 0x message format specifying the following information:

- The address of the writer
- The address of the fee recipient
- The amount of *base token* the writer is offering
- The amount of *quote token* to be paid as a premium to the writer upon purchase
- The expiration time for the sale of this option
- The address of the *CoveredOption* contract for the option they want to write. This address is specified in the taker field of the message, so only the *CoveredOption* contract can take the trade

The writer must have at least as much *base token* as offered, and must set allowance on the *Proxy* contract. Buyers can buy less than the amount of options offered by the writer. In 0x terminology, the writer will be the maker of the trade, and the *CoveredOption* contract will be the taker of the trade. The message can be published in any channel, but is a binding agreement to offer the specified sale. Relayers can then list these option sale offers on an option issuance order book (much the same as relayers in the 0x protocol).

When a buyer wants to purchase an option, they send a transaction to the *CoveredOption* contract that includes the message signed and broadcast by the writer, and the amount of options they wish to buy. Options are issued on a 1:1 ratio with the amount of *base token* deposited by the writer. Once the *CoveredOption* contract receives this transaction it does the following:

1. Validates the expiration date of the option has not yet passed
2. Calls into the *Proxy* to transfer the appropriate amount of *quote token* from the buyer to the *CoveredOption* contract itself. This is the premium that is being paid for the option.
3. Call the *0x Exchange Contract* to exchange the *quote token* which was just taken from the buyer with the appropriate amount of *base token* from the writer. The *0x Exchange Contract* validates the the writer’s signature, ensuring this offer is legitimate. The writer is the maker and the *CoveredOption* contract is the taker in this trade. After this, the writer ends up with the *quote token* premium, and the *CoveredOption* contract ends up with the offered amount of *base token*. The *CoveredOption* contract will hold the *base token* until the option is settled.
4. The *CoveredOption* contract records that the writer has deposited the amount of *base token*. This amount is used later in the case the option expires without being exercised.

5. The balance of the buyer is increased by the amount of options purchased. The buyer is now the holder of that amount of the options, and can now freely transfer and trade them as per the ERC20 standard.
6. If the amount of options available to be written was less than the amount desired by the buyer, the excess *quote token* left over after the trade is transferred back to the buyer.

All of the above steps happen atomically (i.e. they all happen, or none of them happen) in a single transaction.

3.2.4.3 Exercise

Before the option expires, any holder of the option can exercise any amount up to the number of options owned. This means the holder agrees to pay the strike price (globally specified on the *CoveredOption* during its creation), for every option exercised. It is only in the holder's economic interest to exercise the options if the market price for the *base token* is greater than the strike price of the option.

In order to exercise, the owner sends a transaction to the *CoveredOption* contract indicating how many options are to be exercised . Assuming the transaction is valid, the *CoveredOption* contract:

1. Calls into the *Proxy* to transfer $strike\ price \times \#\ options$ of *quote token* from the sender to the *CoveredOption* contract itself
2. Deducts balance from the owner
3. Sends the owner *base token* on a 1:1 basis with number of options exercised
4. Holds onto the *quote token*. The appropriate portion can later be withdrawn by each writer of the option

3.2.4.4 Withdrawal

After the option expires, any writer of the option can withdraw a proportion of both *base token* and *quote token* held by the *CoveredOption* contract corresponding to:

$$\frac{Options\ Written}{Total\ options\ Written} \times (total\ tokens\ held)$$

This is done by sending the *CoveredOption* contract a withdraw transaction, which causes the contract to send the writer their full balance of each token, and sets the writer's written balance to zero.

If an address is both the writer and holder of an equivalent number of options, it may at any time withdraw any amount of *base token* less than or equal to:

$$min(\#\ options\ written,\ \#\ options\ held)$$

Doing so will decrease both the address’s balance and number of options written by the amount withdrawn. This is provided as a utility so a writer can always get the *base token* back, even before the option expires, by purchasing the desired number of options.

4 Governance

Governance will initially be handled by a multisig contract whose keys are held by reputable individuals with a vested interest in the success of dYdX. The powers of this contract will be limited to putting the dYdX protocol into a close-only mode, preventing the creation of any new positions. The contract will have no power to influence any open positions, nor will the contract be able to add new functionality to the protocol. A lack of centralized power is essential to the trustlessness of the protocol. The limited power to put the protocol into close-only mode is intended to be used only to protect would-be users in the event that a major security bug is found.

dYdX enables anyone to increase the functionality of the protocol by allowing users to specify their own smart contracts to help open, close, or manage positions. In this way, any upgrades are completely opt-in by users of the protocols themselves and can also be written by anyone, requiring no special permissions from the base protocol.

In this way, upgrades cannot be forced by the authors of the protocol. Therefore, a token is not currently needed for governance. In the future, to help promote common standards, dYdX will consider using a DAO to govern upgrades to the protocol, however no viable DAOs currently exist.

5 Summary

- dYdX
    - Decentralized protocols for peer-to-peer derivatives and margin trading
    - Built on Ethereum and 0x
    - Open-source and free to use
    - Efficient markets are enabled using off-chain 0x orders and economic incentives for price discovery
    - Modular, extensible smart contracts allow continuous opt-in upgrades
- dYdX Margin Trading Protocol
    - Can be used to profit on downward price movements, or increase leverage
    - Providing low risk fully collateralized loans for margin trades can provide interest fee on long positions
    - Anyone can margin trade or lend any ERC20 token
- dYdX Options Protocol
    - Can be used to reduce risk or speculate
    - Anyone can create, write, buy, or trade any option on any ERC20 token
    - Each option is represented by its own ERC20 token to allow easy trading

6 Acknowledgments

We would like to thank our mentors, advisors, and friends who have provided invaluable advice on dYdX. In particular, we would like to thank Olaf Carlson-Wee, Fred Ehrsam, Linda Xie, and Julian Borrey for reviewing and providing feedback on this work. We would also like to thank Albert Zhou for his help educating us about derivatives. dYdX would not have been possible without their and others' support.

keep-network / whitepaper

Code Issues 18 Pull requests 1 Actions Projects Security Ir

master

whitepaper / keep.tex

Shadowfiend Properly show date last updated … ✓ History

3 contributors

Raw Blame

838 lines (651 sloc) | 35.3 KB

```
%% Based on the style files for ACL-2015

\documentclass[11pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[hidelinks]{hyperref}
\usepackage{acl2015}
\usepackage[numbers]{natbib}
\usepackage{fancyhdr}
\usepackage{textcomp}

\title{The Keep Network:\protect\\A Privacy Layer for Public Blockchains\\
  \vspace*{0.1cm}\textnormal{\normalsize Last updated October 15, 2017}}

\author{Matt Luongo \\
  {\tt mhluongo@gmail.com} \\\And
  Corbin Pon \\
  {\tt corbin.pon@gmail.com} \\}

\pagestyle{fancy}
\fancyhf{}
\renewcommand{\headrulewidth}{0pt}

\lfoot{Draft: \href{https://github.com/keep-network/whitepaper/tree/COMMIT}{COMMIT} - Last updated

\begin{document}

\thispagestyle{fancy}
```

```

\maketitle

\begin{abstract}

  We introduce the keep, a new privacy primitive for developing smart
  contracts on public blockchains, enabling secure storage and usage
  of secrets, as well as supporting infrastructure, including the keep
  market and token.

  Our incremental approach to privacy infrastructure can be brought
  to market on the Ethereum public network, iterated on, and adapted
  for other public blockchains and cross-blockchain use.

\end{abstract}

\section{Motivation}

\subsection{The irony of public blockchains}

Public blockchains have brought unprecedented transparency and
auditability to financial technology. Records are immutable,
verifiable, and censorship-resistant.

Unfortunately, these strengths are also weaknesses for many potential
users.

For every financial use case a public blockchain enables, its public
status restricts another. Bitcoin is touted as a more private payment
method than the traditional financial system, but those familiar with
the technology know that while it may be censorship-resistant, it's
certainly not private by default \cite{bitcoinPrivacy}. Developers
introduced to Ethereum quickly learn to adjust their expectations
\cite{ethereumStackexchange}- all contract state is published to the
blockchain, and can be easily read by competing interests.

These issues are recognized by developers of the Bitcoin and Ethereum
projects.

Confidential transactions \cite{confidentialTransactions} is an
ongoing effort to bring better privacy, and therefore fungibility, to
Bitcoin via sidechains \cite{confidentialTransactionsElements}. The
Zerocash project \cite{zerocash} applied zero-knowledge proofs to
Bitcoin, leading to the creation of Zcash \cite{zcash}, a

cryptocurrency using zk-SNARKs to ensure transaction privacy.

As early as December 2014, Vitalik Buterin, one of the founders of
Ethereum, explored solving this problem with secure multi party
```

```
computation (sMPC) \cite{secretSharingDaos}. In more recent writing,
Buterin shares that ``when [he] and others talk to companies about
building their applications on a blockchain, two primary issues always
come up: scalability and privacy'' \cite{privacyOnTheBlockchain}.

Scalability of public blockchains is a hurdle to mainstream adoption.
Some of the best minds in the cryptocurrency space \cite{lightning}
\cite{ethereumSharding} \cite{plasma} are working on multiple
order-of-magnitude improvements. Privacy, however, hasn't garnered the
same attention, especially in smart contracts.

Basic use cases of smart contracts, including publishing secrets after
certain criteria are met, assessing borrower risk for a loan, and
signing messages and transactions, are incredibly difficult on today's
public blockchains.

\subsection{Existing approaches}

In practice, developers have found a number of ways to build
decentralized applications that use private data.

\subsubsection{The hash-reveal pattern}

A common pattern on public blockchains is to keep private data with
the application's users. Contracts can receive and manipulate hashes
of private data, more generally called commitments
\cite{commitmentScheme}, while users withold the original until
revealing the private data off-chain. We call this the ``hash-reveal''
pattern.

For many applications, this approach is satisfactory. There's a clear
privacy benefit over typical web applications- no centralized
third-party database is at risk. Spreading storage across many users
means a distributed, diverse target for attackers.

There are significant downsides, however. The hash-reveal pattern
requires that all users party to a transaction be online, monitoring
the system, providing private data when necessary, and validating
hashes against private data provided by other users.

This requirement makes the hash-reveal pattern inflexible for complex
protocols, and unsuitable for systems that don't revolve solely around
active human participants, like decentralized autonomous organizations
(DAOs).


\subsubsection{Private blockchains}

Another response to privacy restrictions, primarily from the finance
```

```
industry, has been to build private blockchains, or so-called
``permissioned ledgers''.

These systems operate in a trusted or semi-trusted manner. Instead of
using proof-of-work or other consensus mechanisms designed with an
adversarial network in mind, they can use systems like RAFT to reach
consensus.

One such system, J.P. Morgan's Quorum \cite{quorum}, is a fork of
Ethereum supporting private contract state and messaging between
network participants. Another, Microsoft's recently announced Coco
Framework \cite{coco}, provides data privacy atop an existing private
blockchain.

These systems solve privacy at the expense of many of the benefits of
a public blockchain- trustlessness, public accountability,
censorship-resistance, and permissionless innovation.

\subsubsection{Zero-knowledge proofs}

Zero-knowledge proofs have been leveraged to maintain privacy on
public blockchains- most famously, by the Zcash \cite{zcash} project.

Zero-knowledge proofs allow one party, the prover, to prove a
statement to another party, the verifier, without revealing the
knowledge used to prove that statement.  For example, a prover could
show that they have access to a private key by encrypting a message
chosen by a verifier. The proof can be trivially checked by the
verifier by decrypting the cyphertext with the public key. The private
key, however, remains secret.

More relevant to the domain, zero-knowledge proofs can be used for a
party to prove they have access to funds, or in the case of Zcash, for
a party to prove to miners that a transaction is valid according to
the consensus rules of the network.

Zero-knowledge proofs can be used to construct private financial
systems on a public blockchain. On their own, however, they stop short
of allowing safe delegation of private data from one party to another,
and suffer the same always-online requirements of the ``hash-reveal''
pattern.

Zero-knowledge proofs are a powerful cryptographic tool, and can be
used in conjunction with other techniques to safely delegate secret

access and computation (see section \ref{sMPC}).

\subsection{Public applications, private data}

```

```
None of these techniques adequately address how to build a publicly
verifiable, decentralized, censorship-resistant application that makes
use of private data.

Consider contracts to reveal a secret in case of a dispute between two
parties, to sign a message verifying contract identity off-chain, or
to securely encrypt files \footnote{We go over applications in more
depth later in section \ref{applications}}.

\section{Introducing keeps}

To solve this mismatch between the transparency of public blockchains,
and the need of many autonomous smart contracts for private data, we
introduce the {\em keep}.

A keep is an off-chain container for private data. Keeps allow
contracts to manage and use private data without exposing the data to
the public blockchain.

\subsection{Keep operations}

\begin{table*}[t]
  \centering
  \begin{tabular}{|rp{10cm}|}
  \hline
  \multicolumn{2}{|c|}{\textit{Keep operations}} \\
  \textbf{Create:} & $Contract_{owner}$ publishes a creation request,
  including an initial deposit and a public key,
  $K_{Contract_{owner}}$.\\
  \textbf{Accept:} & A keep, $Keep_{accepted}$, publishes one or more
  public keys $K_{Keep_{accepted_i}}$ signalling readiness.\\
  \textbf{Populate:} & $Contract_{owner}$ publishes an initial
  secret on-chain, encrypted in total or in shares by one or more
  $K_{Keep_{accepted_i}}$, or a specification for a secret to be
  generated.\\
  \textbf{Grant:} & $Contract_{owner}$ publishes another contract
  address, $Contract_{delegate}$, and a permission level, $P_{read}$
  or $P_{admin}$.\\
  \textbf{Compute:} & $Contract_{owner}$ or $Contract_{delegate}$
  publishes a function to compute over the secret, $F(S,...)$, as well
  as other arguments to $F$. Initially $F {\in}
  \{f_{identity},f_{rsa},f_{ecdsa}\}$, though additional functions are
  planned.\\
  \textbf{Results:} & $Keep_{accepted}$ publishes the results

  of its computation, either in whole or in part, over one or more
  invocations.\\
  \textbf{Shutdown:} & $Contract_{owner}$ or $Contract_{delegate}$
  with permission $P_{admin}$ publishes a shutdown request.\\
  \hline
\end{tabular}
\end{table*}

Though keeps maintain state off-chain, they are provisioned and
messaged by contracts on-chain. We will describe the keep in terms of
these on-chain operations. The practical implementation of keeps,
including security guarantees, is covered in sections \ref{eliminatingRisk}
and \ref{keepProviders}.

\subsubsection{Creation and population}

A contract, $Contract_{owner}$, requests a keep by publishing a
request to the blockchain. Once a keep, $Keep_{accepted}$, has accepted a
request and finished initializing off-chain, it will respond to the request
with a set of public keys the calling contract can use to communicate privately
with the keep.

Once the keep has been created, it can be populated in a number of
ways. dApps can publish secret data to the blockchain, encrypted by
the keep's public keys, or send the data to the keep off-chain.
Alternatively, a keep can self-populate with pseudorandom data.

\subsubsection{Publishing data on-chain}

The purpose of a keep is to compute a function over its secret and
publish the results to the blockchain.

Initially, keeps will support publishing their secrets on-chain,
unmodified or encrypted with a public key provided by
$Contract_{owner}$. This enables functionality that's difficult in
today's public smart contracts, like a secret-exposing dead man
switch, useful in a variety of decentralized market schemes.

Keeps can be extended to use their secret in a variety of other ways,
including as key material for symmetric encryption and signing.

\subsubsection{Access management}

The owning contract $Contract_{owner}$ of a keep can delegate access
to the keep to other contracts.

Read and admin access can each be granted, allowing another
contract i($Contract_{delegate}$) to request that a keep's content be

published (read permission, $P_{read}$), or to delegate further access
to other contracts (admin permission, $P_{admin}$). Owners
($Contract_{owner}$) can also revoke their own access.

```

```
Access management enables multi-party secret escrow and auditability
of secret access.

\subsubsection{Destruction}

Depending on the use case, keeps can be long- or short-lived.
Contracts can request that a keep shut down, and should also handle
keeps that are terminated unexpectedly, scenarios which are covered in more
detail later in section \ref{uptime}.

\section{Eliminating third-party risk} \label{eliminatingRisk}

We've described a simple black box for off-chain data storage. The
standardization of this method of secret management will enable
secrets to be bought, sold, and transferred on a public blockchain,
but doesn't inherently solve third-party risks.

Next, we'll describe techniques to eliminate third-party risk.

\subsection{Secure multi party computation} \label{sMPC}

Secure multi party computation (sMPC) is a type of cryptographic
system where a computation is distributed across multiple
participants, some of which may be dishonest. Each participant is
initially given access to a share of a secret by a dealer, and
computes a function over that share. The outputs are then reported to
the dealer, who can assemble the final output, without any participant
learning more than their initial secret share.

Intuitively, sMPC works like this:

\begin{enumerate}
  \item A dealer $D$ wants to compute a function $F$ over a secret,
      $S$.
  \item The dealer selects $n$ parties to the computation, sending
      each of them a share of the secret, $s_i$.
  \item Each party computes a function over their share $f_i(s_i)$ and
      reports the result to the dealer.
  \item The dealer combines these outputs, such that
      $G(f_1(s_1),f_2(s_2),...f_n(s_n)) = F(S)$
\end{enumerate}

The shares $s_i$ should be chosen in such a way that exposing any
share does not jeopardize the secret $S$. A common approach is to use

Shamir's secret sharing \cite{shamir}, such that details about the
secret remain confidential in the face of $n-1$ dishonest parties.

This explanation holds for all $F$ including addition, subtraction,
and multiplication by a known constant. To achieve general
computation, however, we also need to be able to multiply secrets
securely.

Multiplication adds what the literature calls ``rounds''- communication
between the parties, rather than just the dealer $D$.

To multiply two secrets, each party $P_i$ of the $n$ chosen by the
dealer splits its share, $s_i$, into two components, $s_{i1}$ and
$s_{i2}$. The party multiplies those two components, resulting in
$s_{i'}$. Each $P_i$ then acts as a dealer among the the remaining
parties, splitting $s_{i'}$ into $n-1$ pieces.

Each $P_i$ can now resolve their resulting share of the round of
multiplication, $s'_i$, given their access to $n-1$ shares of
$s_{i'}$.

With addition and multiplication, sMPC can securely execute general
computation, at the expense of communication overhead between the
computing parties.

\subsection{sMPC and the blockchain}

sMPC was originally conceived in 1982 \cite{yao1982protocols}, but its
practical application has been limited due to restrictions on the
security model. Existing sMPC solutions only maintain security in the
face of an honest majority of parties.

The advent of the blockchain enables secure usage of sMPC in
adversarial scenarios. By using a public blockchain as an immutable
ledger, sMPC can be made secure in the face of a dishonest
supermajority \cite{spdz}, and, with the requirement of a network
token, can be made strongly Sybil-resistant
(see section \ref{incentivizingProviders}).

For these reasons, sMPC and blockchains are a natural fit. In the
smart contract space, sMPC has been proposed before as a privacy
mechanic.

In 2014, Vitalik Buterin gave a strong introduction to the subject in
an early blog post on privacy on the Ethereum public blockchain
\cite{secretSharingDaos}. In 2016, a team from UMD designed Hawk
\cite{hawk}, a system that marries public and private smart contracts
via sMPC, and the Enigma project out of MIT describes a system related

to ours \cite{enigma}, with a wider focus on general private
computation.

The Keep network will incorporate these ideas into the first
```

```
production-ready sMPC system for a public blockchain.

\section{Keep providers} \label{keepProviders}

The Keep network includes a number of different provider types, each
with their own strengths and tradeoffs. The most important provider,
however, is a novel application of secure multi party computation.

\subsection{Simple sMPC}

Simple sMPC keeps are backed by $n$ nodes, each of which maintain a
share of the provided secret, such that the secret can't be
reconstructed without all $n$ nodes colluding.

These keeps can be populated securely by divvying up a secret into
shares via Shamir secret sharing \cite{shamir}, and encrypting each
share with its respective node's public key. The encrypted shares can
then be published to the public blockchain, or communicated off-chain.

The only computation these keeps will run is an implementation of
distributed RSA \cite{mauland2009realizing} on sMPC, used to publish
encrypted data to the blockchain.

\subsection{Signing sMPC}

The next provider will extend the sMPC keep with two new operations-
securely generating pseudorandom numbers, and signing and encrypting
data, using the keep's contents as a key.

In addition to simple pseudorandom numbers, signing keeps will be able
to generate RSA \cite{mauland2009realizing} and Bitcoin
\cite{gennaro2016threshold,coinparty} key pairs, or be populated with
them via secret sharing.

This means signing keeps will be able to sign and secure contract
communications on- and off-chain, as well as sign transactions for
Bitcoin, Ethereum, and other cryptocurrencies.

Finally, signing keeps can act as pRNG oracles, significantly
improving current methods of random number generation on public
blockchains.

\subsection{Future providers}


The off-chain keep pattern is flexible enough to include a variety of
other pluggable providers, each with their own unique benefits.

\subsubsection{Secure hardware}
```

```

Keeps backed by secure hardware can be used to lower the cost of
securing private data by verifying that only signed code is run
against privileged data.

Instead of requiring $n$ nodes to safely split a secret, a secret can
be sent to a single node that's properly responded to a challenge,
proving it's running signed code. Not only are fewer nodes required,
but these keeps wouldn't suffer the computation overhead of secure
multi-party computation.

This sort of security is fundamentally weaker than that provided by
secure multi-party computation. If a single secure hardware
manufacturer is compromised, it puts all nodes using that hardware at
risk, shifting the threat model. The cost and benefit of this approach
will depend on the application.

\subsubsection{Private smart contracts}

Unlike related work on systems like Enigma \cite{enigma} or Hawk
\cite{hawk}, which use sMPC to build off-chain and alternative-chain
computation networks for private smart contracts, we've chosen to
restrict the initial sMPC keeps to generating, securing, storing,
encrypting, and transmitting secrets. Such restrictions help to
minimize the attack surface on keeps in a production network.

In later work, sMPC schemes can be used to build more feature-rich
keeps. These keeps will enable complex use cases, like operating
private ledgers against public blockchains, or running third-party
code trustlessly on private data.

\section{Incentivizing keep providers} \label{incentivizingProviders}

Providers need to be incentivized to maintain capacity on the network.
Running and securing keeps should be a profitable way to use excess
compute and storage resources.

Consumer contracts, on the other hand, need keeps that will provide:

\begin{itemize}
  \item High availability
  \item Robustness against data loss
  \item Maintenance of confidentiality
  \item Data integrity

\end{itemize}

\subsection{Paying for keeps}

```

```
The best payment structure for keep providers will reward highly
available keeps, and punish poor performance.

<sequence diagram of deposit + per-operation payment>

The two primary costs providers incur are storage and compute, which
map naturally to paying keeps per block and per operation.

Payment per block can be accomplished via a deposit to the managing
contract at the time of keep initialization, metered out over the
lifetime of the keep, and refilled occasionally by the calling
contract. Though this seems like a good fit for payment channels,
minimizing on-chain fees, the security ramifications differ from
typical two-party channels. These differences are discussed further in
the next section.

Payment per operation is simpler. Each request to publish a keep's
contents will require payment of an amount agreed to at the
initialization of a keep.

\subsection{Concerns with uptime and reliability} \label{uptime}

Because availability is vital to using keeps in practice, improper
termination must be disincentivized.

<proper shutdown protocol>

Any keep that doesn't respond properly within a certain block count
threshold to a request will be considered aborted. Aborted keeps will
forfeit all client deposits that have yet to be disbursed. To avoid
skewing client incentives, the deposits that have been earned, but not
yet disbursed, will be burned, and the unearned deposits will be
returned to the client.

Volatility in the crypto currency markets can provide a strong
incentive for a keep provider to improperly terminate a keep. If the
value of the paid currency drops significantly relative to the cost of
running a keep, it's in a provider's best interest to devote their
limited resources to a better-paying client.

To counter this issue, keep providers will need a protocol to
optionally re-negotiate fees for a running keep.

\subsection{Concerns with active attacks}

\label{activeAttacks}

Existing open-source sMPC frameworks, such as VIFF \cite{viff}, are
secure against active attacks in the presence of a \textthreequarters\
supermajority of honest nodes. In such an attack, keeps can be forced to return
malformed data, but secrets can't be compromised unless all nodes with a unique
share backing an sMPC keep are colluding- an extremely high bar for a Sybil
attack.

Recent approaches using SPDZ proofs \cite{spdz} anchored on the
blockchain \cite{bitcoinSmpc, blockchainMultipartyComputation} make
such correctness attacks impossible, even if all nodes backing a keep
are compromised. sMPC keeps will publish proofs to the public
blockchain that can be used to verify correctness. The threat of
active attacks is then reduced to disrupting keep availability, rather
than returning malformed data.

We address the issue of network disruption by introducing two
incentives to keep providers, making active attacks on data
availability impractically expensive.

First, keep providers will be required to prove their holdings in a
token native to the system. Significant disruption of the network
should lead to a drop in the value of the token, incentivizing
provider honesty, lest they devalue their holdings. This scheme also
provides resistance to Sybil attacks--- an active attacker would need to
obtain an outsize portion of all tokens locked up by keep providers to
ensure their overwhelming selection backing new keeps.

Second, keep redundancy can be used to further minimize availability
disruptions \cite{blockchainMultipartyComputation}. All nodes can be
required to include a deposit when they publish their results. If
their results can't be verified by the included SPDZ proof, their
deposit is forfeit to competing nodes.

\section{High-level network design}

Deploying sMPC-based privacy on a public blockchain requires
supporting infrastructure. To build a functional privacy network
against Ethereum, our first target blockchain, we'll introduce
components to ensure fair keep node selection, report results, and
incentivize network actors.

\subsection{The Keep network token}

The native network token, \textit{KEEP}, will be required for
providers to participate.


To be chosen to provide a node for a new keep, a provider must lock up
a minimum stake in KEEP tokens, using a shared staking contract.

At any time, a provider can choose to retrieve their stake--- for
```

```
example, to liquidate their position. All withdrawals, however, will
be subject to a two-week waiting period to disincentivize providers from
quickly staking and withdrawing their position, which could have
adverse effects on running keeps and fair keep selection.

Requiring a native token, rather than the underlying blockchain's
currency, means providers will suffer from negative externalities in
the presence of malicious behavior (see section \ref{activeAttacks}). This sort
of staking also strengthens the system against Sybil attacks (see section
\ref{fairKeepSelection}).

\subsection{Ensuring fair keep selection}
\label{fairKeepSelection}

Contracts requesting keeps and keep providers need to be matched. An
ideal system would enable price discovery, incentivizing new providers
to join if capacity is low, across different keep types.

This matching problem is a great fit for a market. Unfortunately,
on-chain markets are a difficult problem, prone to complexity, miner
frontrunning, and orderbook manipulation. A clever attacker could
manipulate a market, giving them an unfair advantage to be chosen for
a particular keep. Essentially, a two-sided market would expose the
network to Sybil attacks.

In lieu of a market, we need a fair keep selection mechanism.

\subsubsection{Random beacons}

The best way to select providers for a new keep is with a fair coin
toss. Unfortunately, Ethereum only supports deterministic functions.
Contracts that require a random number often rely on a trusted oracle.

A system is only as decentralized as its most centralized component.
Relying on a trusted third party for such a core function of the
project isn't an acceptable risk.

Instead, we can utilize our keep providers as a decentralized source
of entropy. All staked providers can be required to take part in the
random number generation process.

There are a few design considerations for such a system:
\begin{itemize}
  \item Providers can't be allowed an unfair advantage over each other

      in the node selection process.
  \item Each block on the public chain will require at least one
      random number of sufficient size. Today's Ethereum block time is
      25 seconds, but that will likely change significantly in the
      future. The RNG process needs to be fast enough to support much
      shorter block time, if necessary.
  \item RNG needs to be resilient to node failure. Failure in
      production means no new keeps can be created, so resilience
      to partitions between providers as well as against active denial
      of service attacks is desirable.
  \item While not a hard system requirement, providing the Ethereum
      network with a trusted source of randomness will also be a great
      boon to other projects.
\end{itemize}

Most distributed key generation schemes are too slow or prone to
manipulation to be considered. Any scheme we choose should provide
good performance, regardless of the number of participating providers.
Instead, most generation schemes require rounds of communication
between participants, slowing down the key generation process and
providing a large surface for communication failure.

Fortunately, the Dfinity team has solved these issues with their
random beacon design, based on a concept they call threshold relay
\cite{thresholdRelay}.

\subsubsection{Threshold relay}

\begin{table*}[t]
  \centering
  \begin{tabular}{|rp{10cm}|}
  \hline
  \multicolumn{2}{|c|}{\textit{Iterative threshold signatures for
  randomness on existing chains}} \\
  \textbf{Registration:} & As providers join the network, they
  register with at least one threshold group $G_i$ of all groups $G$,
  generating a share of the group's private key, $s_i$. Threshold groups are
  capped at $c$ members, and may intersect. Groups that have reached
  this maximum size publish their public key to the blockchain. We'll
  designate such groups as $G_{registered}$. \\
  \textbf{Trusted setup:} & A trusted party posts a random value
  $r_0$ to the blockchain as the beacon's first output. \\
  \textbf{Bootstrapping:} & $mod(r_{0}, |G_{registered}|)$ is
  used to select a registered threshold group, $G_i$, from
  $G_{registered}$. $G_i$ signs $r_0$ and publishes the
  result, $r_1 = threshold(r_0, s_{0\rightarrow{t}})$ where
  $s_{0\rightarrow{t}}$ is the minimal shares necessary for the group
  to produce a signature. Note that $threshold(...)$ must be a

  deterministic signature scheme to avoid share withholding attacks
  leading to a biased output. \\
  \textbf{Iteration:} & Each block published on the chain will include
  a signature from $G_{registered}$ of the random value $r_i$. As the
  chain grows, the signing threshold groups will change based on
  provider availability. If any group is non-responsive up to its
  threshold $t$, the group is removed from $G_{registered}$. \\
  \textbf{Failure:} & Each iteration is an opportunity for a
  group to fail to generate a valid signature. If a group $G_i$ fails
  to sign the last iteration's random value, $G_{i+1}$ will be used
  instead. \\
  \hline
\end{tabular}
\end{table*}

This work relies on the idea of threshold secret sharing schemes---secret
sharing schemes that retain confidentiality up to some threshold $t$ of
honest actors.

Threshold signatures are a related idea. A threshold signature is a
signature across $n$ parties that requires some minimum $t$ actively
participating to sign. It's a similar idea to "multi-sig" as deployed
in cryptocurrencies today.

Traditional multi-sig, however, requires a smart contract on the
blockchain to validate each signature and release funds. Threshold
signature schemes actually require a threshold $t$ to construct a
signature at all, removing a layer of complexity and coordination
between parties.

The use of threshold signatures means a number of participating
signers in a signing group can be unavailable, and the signature will
still succeed in the presence of $t$ functioning signers. This
provides some of our beacon's required resilience in the face of
failing or misbehaving nodes.

If threshold signatures sound familiar, it might be because they're a
core functionality keeps provide. For example, a keep signing a
Bitcoin transaction does so using threshold ECDSA.

A threshold relay is a way to chain threshold signatures to create a
random beacon. Participants in a threshold relay form threshold
groups. These groups generate new public keys that identify the group
and correspond to a newly generated threshold private key, split
across the participants.

As providers join the network, they will form threshold groups. These
groups will then sign a piece of random data, initially provided by

early network contributors, to bootstrap the relay. The resulting
signature provides the random data for the next iteration, which can
be verified by the rest of the network participants and rejected if
invalid. Each iteration, a new signing group is chosen by the previous
```

```
iteration's random value. As all groups sign the previous iteration's
value, if a signature that's chosen is invalid, the signature from the
next group in line can be chosen instead.

Importantly, the threshold signature scheme needs to be deterministic
to prevent individual shareholders from biasing the signature outcome
in their favor. BLS signatures \cite{BLS} have been used in related
work.

\subsubsection{Keep selection group}

Our threshold relay system will be composed of keep providers seeking
to be chosen to back a new keep, capturing the fees from that keep.

Each block will include a random signature, published by the nominated
keep selection group. Any keeps that require new nodes will have their
providers chosen randomly, using the beacon value from the last block.

In this way, we can ensure fair chances to all staked keep providers,
keeping the cost of a Sybil attack high.

\section{The result registry}

Keeps will offer a number of methods to publish to the public
blockchain. In the case where keeps publish to a smart contract
provided by the keep owner, coordination is simple. In uses that don't
have a natural contract to communicate with, a result registry will be
provided as a default to simplify keep and owner coordination.

\section{Applications}
\label{applications}

\subsection{Dead man switch}

A dead man switch is a device that is automatically activated in case
its owner becomes incapacitated. Keeps enable a particular kind of
dead man switch- publishing a secret, under certain contract
conditions.

Examples of dead man switch applications with keeps include automated
inheritance (``send my beneficiary my private key if I don't check in
quarterly''), arbitration with time limits (``if no decision is made in
10 blocks, publish a shared secret''), as well as protection for
leakers (``publish a key to these insurance files if I don't check

in'').

\subsection{Marketplaces for digital goods}

```

```
Buying and selling digital goods on public blockchains today requires
settling off-chain. Keeps make marketplaces for digital goods, like
audio and video files, straightforward.

Without keeps, each transfer of a private digital good requires one or
more hash-reveal constructions on-chain. More complex scenarios
that require escrow, arbitrators, and other parties who might need
access to the transfered digital good will need ${n^2}$ on-chain
transactions to maintain security. They also require each party to be
online to participate.

Keeps obviate always-online requirements, and simplify the hash-reveal
protocol to access management. All keep access is auditable, and
participants can have access to a keep without viewing its contents,
allowing further optimization.

Without an always-online requirement or complex reveal protocols,
keeps can efficiently support services like iTunes on the blockchain.

\subsection{Pseudorandomness oracle}

Since keeps can populate themselves with random data, they can act as
pseudorandomness oracles, improving on currently popular methods
\cite{prngStackexchange}. sMPC and other secure keeps are a good fit for
decentralized lotteries and other games of chance, as well as offering
a building block for other on-chain algorithms that require
tamper-resistant pRNG.

This capability is an important component of advanced keep uses, like
decentralized signing.

\subsection{Decentralized signing service}

Signing sMPC keep providers are able to sign messages, including
blockchain transactions, using a generated or provided private key.

For the first time, contracts will be able to assert their identity
off-chain, without requiring the recipient's awareness of blockchain
state.

Consider a decentralized signing service for Bitcoin transactions. The
service can participate in multi-signature transactions, only signing
transactions that follow a strict set of rules, including daily
spending limits and recipient whitelists.


Other uses for such a service include second-factor authentication,
where a contract can answer a challenge-response protocol based on
rules on the blockchain.
```

```

\subsection{Custodial wallets and cross-chain trading}

As a special case of a signing service, contracts can use keeps to
generate their own cryptocurrency wallets, taking full custody of any
received funds.

For example, a contract can generate a Bitcoin wallet, and sign
Bitcoin transactions in response to receiving assets on the contract's
native blockchain.

\subsection{Encryption service for blockchain storage}

Services like Filecoin \cite{filecoin} and Storj \cite{storj} are
being built to provide cheap, ubiquitous storage, accessible globally,
via smart contracts and traditional storage interfaces.

These services offer few privacy guarantees by default, leaving the
onus of file encryption on users. Keeps can provide a private bridge
to blockchain storage. By generating an AES key at keep initialization
and providing off-chain data access to the keep, smart contracts can
use keeps to secure files stored on decentralized services.

\subsection{Banking on public blockchains}

As more keep providers are developed, more applications that once
required a private blockchain can be built against public networks.

Traditional finance offers many examples. Consider lending, a basic
service provided by most banks.

There are a number of sensitive variables involved in the lending
process. Borrower credit scores are sensitive; risk assessment is
highly competitive; the terms of a loan aren't typically made public.

Keep providers that execute generic private smart contracts can
protect scores and the risk assessment process, while maintaining
auditability and all other benefits of a public blockchain.

\bibliographystyle{unsrt}
\bibliography{references}{}

\end{document}
```

1inch / **1inchProtocol**

1inch Protocol – fully on-chain DeFi aggregation protocol

MIT License

240 stars 114 forks

Star Notifications

Code Issues 31 Pull requests 12 Actions Projects Security

master Go to file

k06a Fixed Mooniswap integration … on Aug 11, 2020 **244**

View code

README.md

1inch on-chain DeFi aggregation protocol

First ever fully on-chain DEX aggregator protocol by 1inch

build unknown coverage unknown built with OpenZeppelin

Integration

Latest version is always accessible at 1split.eth (beta on 1proto.eth)

Start with checking out solidity interface: IOneSplit.sol

How it works

This smart contract allows to get best price for tokens by aggregating prices from several DEXes.

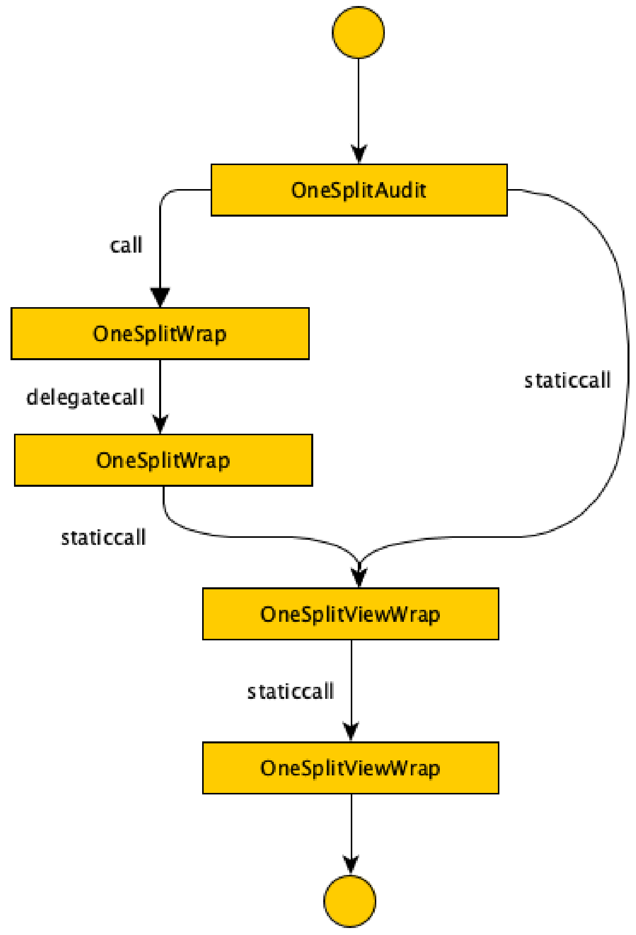
So far the service works with 2 types of exchanges: `split` and `wrap` .

List of `split` exchanges:

```
let splitExchanges = [
    "Uniswap",
    "Kyber",
    "Bancor",
    "Oasis",
    "Curve Compound",
    "Curve USDT",
    "Curve Y",
    "Curve Binance",
    "Curve Synthetix",
    "Uniswap Compound",
    "Uniswap CHAI",
    "Uniswap Aave",
    "Mooniswap",
    "Uniswap V2",
    "Uniswap V2 ETH",
    "Uniswap V2 DAI",
    "Uniswap V2 USDC",
    "Curve Pax",
    "Curve renBTC",
    "Curve tBTC",
    "Dforce XSwap",
    "Shell",
    "mStable mUSD",
    "Curve sBTC",
    "Balancer 1",
    "Balancer 2",
    "Balancer 3",
    "Kyber 1",
    "Kyber 2",
    "Kyber 3",
    "Kyber 4"
]
```

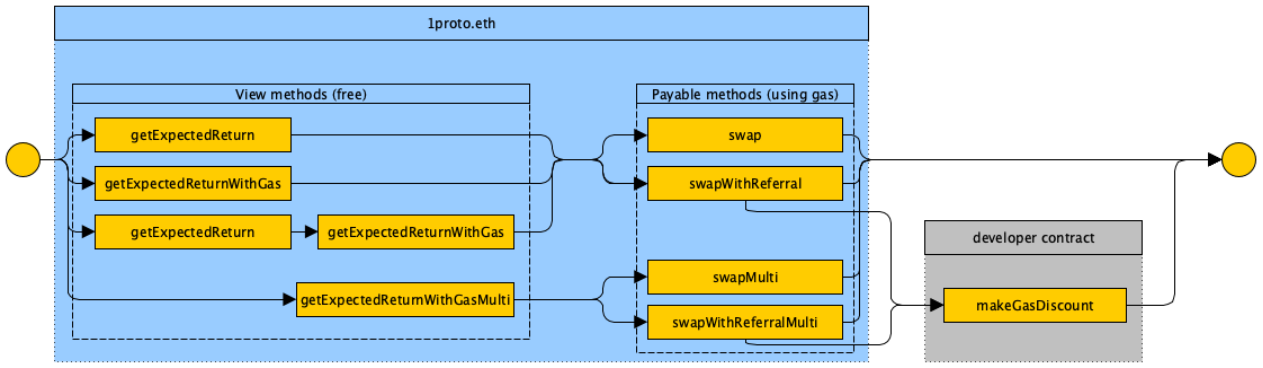
List of `wrap` exchanges:

```
let wrapExchanges = [
    "CHAI",
    "BDAI",
    "Aave",
    "Fulcrum",
    "Compound",
    "Iearn",
    "Idle",
    "WETH",
    "mUSD"
]
```



How to use it

To use this service you have to call methods at OneSplitAudit



To swap tokens you have to figure out way from left to right points by one of paths on scheme above.

For example, first of all call method `getExpectedReturn` (see methods section), it returns `distribution` array. Each element of this array matches element of `splitExchanges` (see above) and represents fraction of trading volume.
Then call `getExpectedReturnWithGas` to take into account gas when splitting. This method returns more profitable `distribution` array for exchange.
Then call method `swap` or `swapWithReferral` (see methods section) with param `distribution` which was recieved earlier from method `getExpectedReturn`.

Swap may be customized by flags (see flags section). There are 2 types of swap: direct swap and swap over transitional token.

In case of direct swap each element of `distribution` array matches element of `splitExchanges` and represents fraction of trading off token as alerady described above.

In case of swap with transitional token each element of `distribution` (256 bits) matches 2 swaps: second bytes are equal to swap to transitional token, lowest bytes are equal to swap to the desired token.

Supported DEXes

- Uniswap
- Uniswap V2
- Kyber
- Bancor
- Oasis
- Curve
- Mooniswap
- Dforce XSwap
- Shell
- mStable
- CHAI
- BDAI
- Aave
- Fulcrum
- Compound
- Iearn
- Idle
- WETH

Methods

If you need Ether instead of any token use `address(0)` or `address(0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE)` as param `fromToken` / `destToken`

getExpectedReturn

```
function getExpectedReturn(
    IERC20 fromToken,
    IERC20 destToken,
    uint256 amount,
    uint256 parts,
    uint256 flags
)
    public
    view
    returns(
        uint256 returnAmount,
        uint256[] memory distribution
    )
```

Calculate expected returning amount of desired token

Params	Type	Description
fromToken	IERC20	Address of trading off token
destToken	IERC20	Address of desired token
amount	uint256	Amount for `fromToken`
parts	uint256	Number of pieces source volume could be splitted (Works like granularity, higly affects gas usage. Should be called offchain, but could be called onchain if user swaps not his own funds, but this is still considered as not safe)
flags	uint256	Flags for enabling and disabling some features (default: `0`), see flags description

Return values:

Params	Type	Description
returnAmount	uint256	Expected returning amount of desired token

Params	Type	Description
distribution	uint256[]	Array of weights for volume distribution

Notice: This method is equal to `getExpectedReturnWithGas(fromToken, destToken, amount, parts, flags, 0)`

Example:

```
let Web3 = require('web3')

let provider = new
Web3.providers.WebsocketProvider('wss://mainnet.infura.io/ws/v3/YOUR_TOKEN')
let web3 = new Web3(provider)

let ABI = [{"inputs":[{"internalType":"contract
IOneSplitMulti","name":"impl","type":"address"}],"payable":false,"stateMutability":"
{"anonymous":false,"inputs":
[{"indexed":true,"internalType":"address","name":"newImpl","type":"address"}],"name"
{"anonymous":false,"inputs":
[{"indexed":true,"internalType":"address","name":"previousOwner","type":"address"},
{"indexed":true,"internalType":"address","name":"newOwner","type":"address"}],"name"
{"anonymous":false,"inputs":[{"indexed":true,"internalType":"contract
IERC20","name":"fromToken","type":"address"},
{"indexed":true,"internalType":"contract
IERC20","name":"destToken","type":"address"},
{"indexed":false,"internalType":"uint256","name":"fromTokenAmount","type":"uint256"}
{"indexed":false,"internalType":"uint256","name":"destTokenAmount","type":"uint256"}
{"indexed":false,"internalType":"uint256","name":"minReturn","type":"uint256"},
{"indexed":false,"internalType":"uint256[]","name":"distribution","type":"uint256[]"
{"indexed":false,"internalType":"uint256[]","name":"flags","type":"uint256[]"},
{"indexed":false,"internalType":"address","name":"referral","type":"address"},
{"indexed":false,"internalType":"uint256","name":"feePercent","type":"uint256"}],"na
{"payable":true,"stateMutability":"payable","type":"fallback"},
{"constant":true,"inputs":[],"name":"chi","outputs":[{"internalType":"contract
IFreeFromUpTo","name":"","type":"address"}],"payable":false,"stateMutability":"view"
{"constant":false,"inputs":[{"internalType":"contract
IERC20","name":"asset","type":"address"},
{"internalType":"uint256","name":"amount","type":"uint256"}],"name":"claimAsset","ou
[],"payable":false,"stateMutability":"nonpayable","type":"function"},
{"constant":true,"inputs":[{"internalType":"contract
IERC20","name":"fromToken","type":"address"},{"internalType":"contract
IERC20","name":"destToken","type":"address"},
{"internalType":"uint256","name":"amount","type":"uint256"},
{"internalType":"uint256","name":"parts","type":"uint256"},
{"internalType":"uint256","name":"flags","type":"uint256"}],"name":"getExpectedRetur
[{"internalType":"uint256","name":"returnAmount","type":"uint256"},
{"internalType":"uint256[]","name":"distribution","type":"uint256[]"}],"payable":fal
{"constant":true,"inputs":[{"internalType":"contract
```

```
IERC20","name":"fromToken","type":"address"},{"internalType":"contract
IERC20","name":"destToken","type":"address"},
{"internalType":"uint256","name":"amount","type":"uint256"},
{"internalType":"uint256","name":"parts","type":"uint256"},
{"internalType":"uint256","name":"flags","type":"uint256"},
{"internalType":"uint256","name":"destTokenEthPriceTimesGasPrice","type":"uint256"}]
[{"internalType":"uint256","name":"returnAmount","type":"uint256"},
{"internalType":"uint256","name":"estimateGasAmount","type":"uint256"},
{"internalType":"uint256[]","name":"distribution","type":"uint256[]"}],"payable":fal
{"constant":true,"inputs":[{"internalType":"contract
IERC20[]","name":"tokens","type":"address[]"},
{"internalType":"uint256","name":"amount","type":"uint256"},
{"internalType":"uint256[]","name":"parts","type":"uint256[]"},
{"internalType":"uint256[]","name":"flags","type":"uint256[]"},
{"internalType":"uint256[]","name":"destTokenEthPriceTimesGasPrices","type":"uint256
[{"internalType":"uint256[]","name":"returnAmounts","type":"uint256[]"},
{"internalType":"uint256","name":"estimateGasAmount","type":"uint256"},
{"internalType":"uint256[]","name":"distribution","type":"uint256[]"}],"payable":fal
{"constant":true,"inputs":[],"name":"isOwner","outputs":
[{"internalType":"bool","name":"","type":"bool"}],"payable":false,"stateMutability":
{"constant":true,"inputs":[],"name":"oneSplitImpl","outputs":
[{"internalType":"contract
IOneSplitMulti","name":"","type":"address"}],"payable":false,"stateMutability":"view
{"constant":true,"inputs":[],"name":"owner","outputs":
[{"internalType":"address","name":"","type":"address"}],"payable":false,"stateMutabi
{"constant":false,"inputs":[],"name":"renounceOwnership","outputs":
[],"payable":false,"stateMutability":"nonpayable","type":"function"},
{"constant":false,"inputs":[{"internalType":"contract
IOneSplitMulti","name":"impl","type":"address"}],"name":"setNewImpl","outputs":
[],"payable":false,"stateMutability":"nonpayable","type":"function"},
{"constant":false,"inputs":[{"internalType":"contract
IERC20","name":"fromToken","type":"address"},{"internalType":"contract
IERC20","name":"destToken","type":"address"},
{"internalType":"uint256","name":"amount","type":"uint256"},
{"internalType":"uint256","name":"minReturn","type":"uint256"},
{"internalType":"uint256[]","name":"distribution","type":"uint256[]"},
{"internalType":"uint256","name":"flags","type":"uint256"}],"name":"swap","outputs":
[{"internalType":"uint256","name":"","type":"uint256"}],"payable":true,"stateMutabil
{"constant":false,"inputs":[{"internalType":"contract
IERC20[]","name":"tokens","type":"address[]"},
{"internalType":"uint256","name":"amount","type":"uint256"},
{"internalType":"uint256","name":"minReturn","type":"uint256"},
{"internalType":"uint256[]","name":"distribution","type":"uint256[]"},
{"internalType":"uint256[]","name":"flags","type":"uint256[]"}],"name":"swapMulti","
[{"internalType":"uint256","name":"","type":"uint256"}],"payable":true,"stateMutabil
{"constant":false,"inputs":[{"internalType":"contract
IERC20","name":"fromToken","type":"address"},{"internalType":"contract
IERC20","name":"destToken","type":"address"},
{"internalType":"uint256","name":"amount","type":"uint256"},
{"internalType":"uint256","name":"minReturn","type":"uint256"},
{"internalType":"uint256[]","name":"distribution","type":"uint256[]"},
{"internalType":"uint256","name":"flags","type":"uint256"},
{"internalType":"address","name":"referral","type":"address"},
{"internalType":"uint256","name":"feePercent","type":"uint256"}],"name":"swapWithRef
[{"internalType":"uint256","name":"","type":"uint256"}],"payable":true,"stateMutabil
{"constant":false,"inputs":[{"internalType":"contract
IERC20[]","name":"tokens","type":"address[]"},
{"internalType":"uint256","name":"amount","type":"uint256"},
{"internalType":"uint256","name":"minReturn","type":"uint256"},
{"internalType":"uint256[]","name":"distribution","type":"uint256[]"},
{"internalType":"uint256[]","name":"flags","type":"uint256[]"},
{"internalType":"address","name":"referral","type":"address"},
{"internalType":"uint256","name":"feePercent","type":"uint256"}],"name":"swapWithRef
[{"internalType":"uint256","name":"returnAmount","type":"uint256"}],"payable":true,"
{"constant":false,"inputs":
[{"internalType":"address","name":"newOwner","type":"address"}],"name":"transferOwne
[],"payable":false,"stateMutability":"nonpayable","type":"function"}]
let CONTRACT_ADDRESS = "0xC586BeF4a0992C495Cf22e1aeEE4E446CECDee0E"

let contract = new web3.eth.Contract(ABI, CONTRACT_ADDRESS)
contract.methods.getExpectedReturn(
    "0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE",
    "0x89d24a6b4ccb1b6faa2625fe562bdd9a23260359",
    100,
    10,
    0
).call().then(data => {
    console.log(`returnAmount: ${data.returnAmount.toString()}`)
    console.log(`distribution: ${JSON.stringify(data.distribution)}`)
}).catch(error => {
    // TO DO: ...
});
```

getExpectedReturnWithGas

```
function getExpectedReturnWithGas(
    IERC20 fromToken,
    IERC20 destToken,
    uint256 amount,
    uint256 parts,
    uint256 flags,
    uint256 destTokenEthPriceTimesGasPrice
)
    public
    view
    returns(
        uint256 returnAmount,
        uint256 estimateGasAmount,
        uint256[] memory distribution
    )
```

Calculate expected returning amount of desired token taking into account how gas protocols affect price

Params	Type	Description
fromToken	IERC20	Address of trading off token
destToken	IERC20	Address of desired token
amount	uint256	Amount for `fromToken`
parts	uint256	Number of pieces source volume could be splitted (Works like granularity, higly affects gas usage. Should be called offchain, but could be called onchain if user swaps not his own funds, but this is still considered as not safe)
flags	uint256	Flags for enabling and disabling some features (default: `0`), see flags description
destTokenEthPriceTimesGasPrice	uint256	`returnAmount * gas_price` , where `returnAmount` is result of `getExpectedReturn(fromToken, destToken, amount, parts, flags)`

Return values:

Params	Type	Description
returnAmount	uint256	Expected returning amount of desired token
estimateGasAmount	uint256	Expected gas amount of exchange
distribution	uint256[]	Array of weights for volume distribution

Example:

```
// TO DO: ...
```

getExpectedReturnWithGasMulti

```
function getExpectedReturnWithGasMulti(
    IERC20[] memory tokens,
    uint256 amount,
    uint256[] memory parts,
    uint256[] memory flags,
    uint256[] memory destTokenEthPriceTimesGasPrices
)
    public
    view
    returns(
        uint256[] memory returnAmounts,
        uint256 estimateGasAmount,
        uint256[] memory distribution
    )
```

Calculate expected returning amount of first `tokens` element to last `tokens` element through and the middle tokens with corresponding `parts`, `flags` and `destTokenEthPriceTimesGasPrices` array values of each step.
The length of each array (`parts`, `flags` and `destTokenEthPriceTimesGasPrices`) should be 1 element less than `tokens` array length. Each element from `parts`, `flags` and `destTokenEthPriceTimesGasPrices` corresponds to 2 neighboring elements from `tokens` array.

Params	Type	Description
tokens	IERC20[]	The sequence of tokens swaps (`tokens[0] -> tokens[1] -> ...`)
amount	uint256	Amount for `tokens[0]`
parts	uint256[]	The sequence of number of pieces source volume could be splitted (Works like granularity, higly affects gas usage. Should be called offchain, but could be called onchain if user swaps not his own funds, but this is still considered as not safe)
flags	uint256[]	The sequence of flags for enabling and disabling some features (default: `0`), see flags description

Params	Type	Description
destTokenEthPriceTimesGasPrice	uint256[]	The sequence of numbers `returnAmount * gas_price`, where `returnAmount` is result of `getExpectedReturn(fromToken, destToken, amount, parts, flags)`

Return values:

Params	Type	Description
returnAmount	uint256[]	Expected returning amounts of desired tokens in `tokens` array
estimateGasAmount	uint256	Expected gas amount of exchange
distribution	uint256[]	Array of weights for volume distribution

Example:

```
// TO DO: ...
```

swap

```
function swap(
    IERC20 fromToken,
    IERC20 destToken,
    uint256 amount,
    uint256 minReturn,
    uint256[] memory distribution,
    uint256 flags
) public payable returns(uint256)
```

Swap `amount` of `fromToken` to `destToken`

Params	Type	Description
fromToken	IERC20	Address of trading off token
destToken	IERC20	Address of desired token
amount	uint256	Amount for `fromToken`

Params	Type	Description
minReturn	uint256	Minimum expected return, else revert transaction
distribution	uint256[]	Array of weights for volume distribution (returned by `getExpectedReturn`)
flags	uint256	Flags for enabling and disabling some features (default: `0`), see flags description

Notice: Make sure the `flags` param coincides `flags` param in `getExpectedReturn` method if you want the same result

Notice: This method is equal to `swapWithReferral(fromToken, destToken, amount, minReturn, distribution, flags, address(0), 0)`

Return values:

Params	Type	Description
returnAmount	uint256	Recieved amount of desired token

Example:

```
// TO DO: ...
```

swapMulti

```
function swapMulti(
    IERC20[] memory tokens,
    uint256 amount,
    uint256 minReturn,
    uint256[] memory distribution,
    uint256[] memory flags
) public payable returns(uint256)
```

Swap `amount` of first element of `tokens` to the latest element.
The length of `flags` array should be 1 element less than `tokens` array length. Each element from `flags` array corresponds to 2 neighboring elements from `tokens` array.

Params	Type	Description
tokens	IERC20[]	Addresses of tokens or `address(0)` for Ether

Params	Type	Description
amount	uint256	Amount for `tokens[0]`
minReturn	uint256	Minimum expected return, else revert transaction
distribution	uint256[]	Array of weights for volume distribution (returned by `getExpectedReturn`)
flags	uint256[]	The sequence of flags for enabling and disabling some features (default: `0`), see flags description

Notice: Make sure the `flags` param coincides `flags` param in `getExpectedReturnWithGasMulti` method if you want the same result

Notice: This method is equal to `swapWithReferralMulti(fromToken, destToken, amount, minReturn, distribution, flags, address(0), 0)`

Return values:

Params	Type	Description
returnAmount	uint256	Recieved amount of desired token

Example:

```
// TO DO: ...
```

swapWithReferral

```
function swapWithReferral(
    IERC20 fromToken,
    IERC20 destToken,
    uint256 amount,
    uint256 minReturn,
    uint256[] memory distribution,
    uint256 flags,
    address referral,
    uint256 feePercent
) public payable returns(uint256)
```

Swap `amount` of `fromToken` to `destToken`

Params	Type	Description

Params	Type	Description
fromToken	IERC20	Address of trading off token
destToken	IERC20	Address of desired token
amount	uint256	Amount for `fromToken`
minReturn	uint256	Minimum expected return, else revert transaction
distribution	uint256[]	Array of weights for volume distribution (returned by `getExpectedReturn`)
flags	uint256	Flags for enabling and disabling some features (default: `0`), see flags description
referral	address	Referrer's address (exception with flag `FLAG_ENABLE_REFERRAL_GAS_SPONSORSHIP`)
feePercent	uint256	Fees percents normalized to 1e18, limited to 0.03e18 (3%)

Notice: Make sure the `flags` param coincides `flags` param in `getExpectedReturn` method if you want the same result

Return values:

Params	Type	Description
returnAmount	uint256	Recieved amount of desired token

Example:

```
// TO DO: ...
```

swapWithReferralMulti

```
function swapWithReferralMulti(
    IERC20[] memory tokens,
    uint256 amount,
    uint256 minReturn,
    uint256[] memory distribution,
    uint256[] memory flags,
    address referral,
    uint256 feePercent
) public payable returns(uint256 returnAmount)
```

Swap `amount` of first element of `tokens` to the latest element.
The length of `flags` array should be 1 element less than `tokens` array length. Each element from `flags` array corresponds to 2 neighboring elements from `tokens` array.

Params	Type	Description
tokens	IERC20[]	Addresses of tokens or `address(0)` for Ether
amount	uint256	Amount for `tokens[0]`
minReturn	uint256	Minimum expected return, else revert transaction
distribution	uint256[]	Array of weights for volume distribution (returned by `getExpectedReturn`)
flags	uint256[]	The sequence of flags for enabling and disabling some features (default: `0`), see flags description
referral	address	Referrer's address (exception with flag `FLAG_ENABLE_REFERRAL_GAS_SPONSORSHIP`)
feePercent	uint256	Fees percents normalized to 1e18, limited to 0.03e18 (3%)

Notice: Make sure the `flags` param coincides `flags` param in `getExpectedReturnWithGasMulti` method if you want the same result

Return values:

Params	Type	Description
returnAmount	uint256	Recieved amount of desired token

Example:

```
// TO DO: ...
```

makeGasDiscount

```
function makeGasDiscount(
    uint256 gasSpent,
    uint256 returnAmount,
    bytes calldata msgSenderCalldata
)
```

In case developer wants to manage burning `GAS` or `CHI` tokens with developer's own smartcontract one should implement this method and use `FLAG_ENABLE_REFERRAL_GAS_SPONSORSHIP` flag. `1proto.eth` will call `makeGasDiscount` in developer's smartcontract.

Params	Type	Description
gasSpent	uint256	How many gas was spent
returnAmount	uint256	Recieved amount of desired token
msgSenderCalldata	bytes	Arguments from `swap`, `swapWithReferral` or `swapWithReferralMulti` method

Notice: There is no such method in `1proto.eth`.

Flags

Flag types

There are basically 3 types of flags:

1. **Exchange switch**
   This flags allow `1split.eth` to enable or disable using exchange pools for swap. This can be applied for exchanges in genereral, for example: `split`, `wrap`, or this can be applied for a specific exchange type, for example: `bancor`, `oasis`.
   This flags may be used in any combination.

2. **Transitional token selector**
   This flags provide to swap from `fromToken` to `destToken` using transitional token.
   This flags cann't be used in combination with the same type.

3. **Functional flags**
   This flags provide some additional features.
   This flags may be used in any combination.

Flags description

`flags` param in `1split.eth` methods is sum of flags values, for example:

```
flags = FLAG_DISABLE_UNISWAP + FLAG_DISABLE_KYBER + ...
```