Loopring:
A Decentralized Token Exchange Protocol

Daniel Wang
daniel@loopring.org

Jay Zhou
jay@loopring.org

Alex Wang
alex@loopring.org

Matthew Finestone
matt.finestone@gmail.com

https://loopring.org

March 15, 2018

Abstract

Loopring is an open protocol for building decentralized exchanges. Loopring operates as a public set of smart contracts responsible for trade and settlement, with an off-chain group of actors aggregating and communicating orders. The protocol is free, extensible, and serves as a standardized building block for decentralized applications (dApps) that incorporate exchange functionality. Important improvements over current decentralized exchange protocols include the ability for users' orders to be mix-and-matched with other orders, obviating the constraints of two-token trading pairs, drastically improving liquidity, and allowing for price improvement possibility. Loopring also employs a robust and unique solution to prevent front-running: the unfair attempt to submit transactions into a block quicker than the original solution provider. Loopring is blockchain agnostic, and deployable on any blockchain with smart contract functionality. At the time of writing, it's operable on Ethereum [1] [2] and Qtum [3] with NEO [4] also under construction. Its interoperable standards ensure trustless, decentralized, and anonymous trading. `text`

1 Introduction

With the proliferation of blockchain-based assets, the need to exchange these assets amongst counterparties has significantly increased. As thousands of new tokens are introduced - including the tokenization of traditional assets - this need is magnified. Whether exchanging tokens for speculative trading motivations, or converting to access networks via their native utility tokens, the ability to exchange one cryptoasset for another is foundational for the larger ecosystem. There is a potential energy in assets [5], and realizing this energy - unlocking capital - requires not only ownership rights, but the ability to freely transfer and transform these assets.

As such, the trustless exchange of tokens (value) is a compelling use case for blockchain technology. Until now, however, crypto enthusiasts have largely settled for trading tokens on traditional centralized exchanges. The Loopring protocol is needed because, just as Bitcoin [6] dutifully pointed out that, in regards to peer-to-peer electronic cash, "the main benefits are lost if a trusted third party is still required to prevent double-spending", so too are the main benefits of decentralized assets lost if they must pass through trusted, gated, centralized exchanges.

Trading decentralized tokens on centralized exchanges doesn't make sense from a philosophical perspective, as it fails to uphold the virtues these decentralized projects espouse. There are also numerous practical risks and limitations in using centralized exchanges which are described below. Decentralized exchanges (DEXs) [7] [8] [9] [10] have sought to address these issues, and in many cases have succeeded in alleviating security risks by using blockchains for disintermediation. However, as DEX capability becomes important infrastructure for the new economy, there is substantial room for improvement in performance. Loopring aims to provide critical tools for said infrastructure with its dApp and blockchain agnostic open protocol.

2 Current Exchange Landscape

2.1 Inadequacies of Centralized Exchanges

The three primary risks of centralized exchanges are; 1) Lack of security, 2) Lack of transparency, and 3) Lack of liquidity.

Lack of Security arises from users typically surrendering control of their private keys (funds) to one centralized entity. This exposes users to the possibility that centralized exchanges fall prey to malicious hackers. The security and hacking risks facing all centralized exchanges are well known [11] [12] [13], yet are often accepted as "table stakes" for token trading. Centralized exchanges, such as Coinbase and Bittrex, continue to be honeypots for hackers to attack,

as their servers control millions of dollars of users' funds. Exchange developers can also make honest, accidental errors with user funds. Users are simply not in control of their own tokens when deposited at a centralized exchange.

Lack of Transparency exposes users to the risk of dishonest exchanges acting unfairly. The distinction here is by the exchange operator's malintentions since users are not truly trading their own assets on a centralized exchange, but rather, an IOU. When tokens are sent to the exchange's wallet, they take custody, and offer an IOU. All trades are then effectively between users' IOUs. To withdraw, you redeem your IOU with the exchange, and receive your tokens to your external wallet address. Throughout this process there is a lack of transparency, and the exchange can shutdown, freeze your account, go bankrupt, etc. It is also possible that they use your assets for other purposes while in custody, such as lending them out to third parties. Lack of transparency can cost users without a total loss of funds, such as in higher trading fees, delays at peak demand, regulatory risk, and orders being front ran.

Lack of Liquidity. From the point of view of exchange operators, fragmented liquidity inhibits entry by new exchanges because of two winner-takes-all scenarios. First, the exchange with the greatest number of trading pairs available wins, because users find it desirable to conduct all their trades on one exchange. Second, the exchange with the largest order book wins, because of favorable bid-ask spreads for each trading pair. This discourages competition from newcomers because it is difficult for them to build up initial liquidity. As a result, many legacy exchanges command a high market share despite user complaints and even major hacking incidents. It's worth noting that as centralized exchanges win market share, they make themselves an ever-larger hacking target.

From the point of view of users, fragmented liquidity significantly reduces user experience. In a centralized exchange, users are only able to trade within the exchange's own liquidity pools, against its own order book, and between its supported token pairs. To trade Token A for Token B, users must go to an exchange that supports both tokens or register at different exchanges, disclosing personal information. Users often need to execute preliminary or intermediate trades, typically against BTC or ETH, paying bid-ask spreads in the process. Finally, the order books may not be deep enough to complete the trade without material slippage.

The result is disconnected silos of liquidity and a fragmented ecosystem that resembles the legacy financial system, with significant trading volume centralized on few exchanges. The global liquidity promises of blockchains hold no merit within centralized exchanges.

2.2 Inadequacies of Decentralized Exchanges

Decentralized exchanges, such as KyberNetwork or EtherDelta, differ from centralized exchanges in part because users maintain control of their assets (private keys) by performing trades directly on the underlying blockchain. By leveraging the trustless technology of cryptocurrencies themselves, they successfully mitigate many of the above-mentioned risks surrounding security. However, problems persist in regards to performance and structural limitations.

Liquidity often remains an issue as users must search for counterparties across disparate liquidity pools and standards. Fragmented liquidity effects are present if DEXs or dApps at large don't employ consistent standards to interoperate, and if orders are not shared/propagated across a wide network. The liquidity of limit order books, and, specifically their resiliency - how fast filled limit orders are regenerated - can significantly affect optimal trading strategies[14].

Furthermore, since trades are performed on chain, DEXs inherit the limitations of the underlying blockchain, namely: scalability, delays in execution (mining), and costly modifications to orders. Thus, blockchain order books do not scale well, as executing code on the blockchain incurs a cost (gas), making multiple order-cancel cadences prohibitively expensive.

Finally, because blockchain order books are public, the transaction to place an order is visible by miners as it awaits being mined into the next block, and placed into an order book. This delay exposes the user to the risk of being front run and having the price or execution move against him.

2.3 Hybrid Solutions

For the above reasons, purely blockchain-based exchanges have limitations that make them uncompetitive with centralized exchanges. There is a tradeoff between on-chain inherent trustlessness, and centralized exchange speed and order flexibility. Protocols such as Loopring and 0x [10] extend a solution of on-chain settlement with off-chain order relay. These solutions revolve around open smart contracts, but navigate scalability limitations by performing several functions off-chain, and giving nodes flexibility in fulfilling critical roles for the network. However, there remain drawbacks[15] for the hybrid model as well. The Loopring protocol proposes meaningful differences in our approach to a hybrid solution throughout the paper.

3 Loopring Protocol

Loopring is not a DEX, but a modular protocol for building DEXs on multiple blockchains. We disassemble the component parts of a traditional exchange and offer a set of public smart contracts and decentralized actors in its place. The roles in the network include wallets, relays, liquidity-sharing

consortium blockchains, order book browsers, ring-miners, and asset tokenization services. Before defining each, we should first understand what a Loopring compliant order looks like.

3.1 Order Ring

Loopring orders are expressed in what we call a Unidirectional Order Model (UDOM [16]). UDOM expresses orders as token exchange requests - amountS/amountB - instead of bids and asks. Since every order is just an exchange rate between two tokens, a powerful feature of the protocol is the mixing and matching of multiple orders in circular trade. By using up to 10 orders instead of a single trading pair, there is a dramatic increase in liquidity and potential for price improvement.

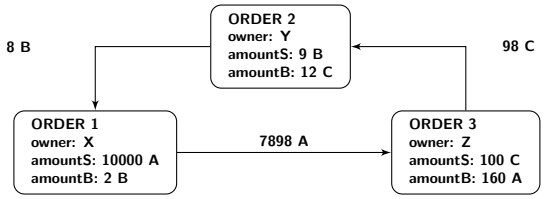

Figure 1: A Ring of 3 Orders

Definition 3.1 (Order Ring) *Let C_0, C_1, $\cdots$, C_{n-1} be n different kinds of token, $O_{0\to1}$, $\cdots$, $O_{i\to i\oplus1}$, $\cdots$, $O_{n-1\to0}$ be n orders. Those orders can form a ring for trading:*

$$O_{0\to1} \to \cdots \to O_{i\to i\oplus1} \to \cdots \to O_{n-1\to0},$$

where n is the length of the ring, and $i \oplus 1 \equiv i+1 \mod n$.

Each order's token to sell is another order's token to buy. It creates a loop that allows each order to exchange their desired tokens without requiring an opposing order for its pair. Traditional order pair trades can, of course, still be executed, in what is essentially a special case of an Order Ring.

An order ring is valid when all component transactions can be executed at an exchange rate equal to or better than the original rate specified by the user. To verify ring validity, Loopring protocol smart contract (protocol) must receive Order Rings from miners where the product of the original exchange rates of all orders is equal to or greater than 1.

Let's assume Alice and Bob want to trade their tokens A and B. Alice has 15 tokens A and she wants 4 tokens B for them; Bob has 10 tokens B and he wants 30 tokens A for them.

Who is buying and who is selling? This depends only on the asset we fix to give price quotations. If token A is the reference, then Alice is buying tokens B for $\frac{15}{4} = 3.75$ XTA, while Bob is selling 10 tokens B for $\frac{30}{10} = 3.00$ XTA. In the case of fixing token B as reference we say that Alice is selling 15 tokens A for $\frac{4}{15} = 0.26666667$ XTB and Bob is buying 10 tokens A for $\frac{10}{30} = 0.33333334$ XTB. Hence, who's the buyer or seller is purely arbitrary.

In the first situation Alice is willing to pay a higher price than the price Bob is selling his tokens for, in the second situation Bob is willing to pay a higher price than the price Alice is selling her tokens for. It is clear that a trade is possible whenever the buyer is willing to pay an equal or higher price than the selling price.

$$\frac{\frac{15}{4}}{\frac{30}{10}} = \frac{3.75}{3.00} = \frac{0.33333334}{0.26666667} = \frac{\frac{10}{30}}{\frac{4}{15}} = \frac{15}{4}\cdot\frac{10}{30} = \frac{150}{120} = 1.25 > 1 \tag{1}$$

Thus, for a set of n orders to be able to be filled, in full or partially, we need to know if the product of each one of the exchange rates as buy orders results in a number greater or equal to 1. If so, all the n orders can be either partially, or totally filled.

In this article, we reference this as the Uni-Directional Order Model, or UDOM for short. See 7.1 for more details about Loopring's orders.

4 Ecosystem Participants

The following ecosystem participants jointly provide all functionalities a centralized exchange has to offer.

- **Wallets**: A common wallet service or interface that gives users access to their tokens and a way to send orders to the Loopring network. Wallets will be incentivized to produce orders by sharing fees (see 8).
- **Relays / Ring Miners**: Relays are nodes that form a decentralized network for order propagation. They maintain public order books and trade history and broadcast new orders to other relays via any arbitrary off-chain medium. Ring-mining is a feature of relays. It is computationally heavy and is done completely off-chain. Ring-mining produces Order Rings: rings of between 2 and 10 tokens created by stitching together disparate orders. Relays are free in how they choose to communicate with one another.
- **Consortium Liquidity Sharing Blockchain**: TODO(daniel)
- **Loopring Protocol Smart Contracts (LSC)**: A set of public and free smart contracts that checks Order Rings received from miners, does token transfers on behalf of users, incentivizes miners/wallets, and emits events. Relays/order browsers listen to these events to keep their order books and trade history up to date. See A for details.
- **Asset Tokenization Services**: A bridge between assets that cannot be directly traded on Loopring. They are centralized services run by trustworthy companies or organizations. A user could deposit his assets (real, fiat or tokens from other chains) and get tokens

issued. By returning these tokens the user gets back his deposit. Loopring is not a cross-chain exchange protocol, but Asset Tokenization Services make it possible to trade Ethereum ERC20 [17] tokens with physical assets as well as assets on other blockchains.

5 Exchange Process

1. **Order Initiation & ERC20 Authorization**: A user wants to exchange X amount of TokenA for Y amount of TokenB. The current rate and order book for this pair can be found on multiple sources provided by relays or other interfaces hooked up to the network, such as order book browsers. The user places an order through a wallet interface. An amount of LRx can be added to the order as a fee for miners; higher LRx fee means a better chance to be processed earlier by miners. The wallet authorizes the LSC to handle X amount of TokenA the user wants to sell, but does not lock the user's tokens, who remains free to move them while the order is being processed.

2. **Send order to the network**: Once the authorization is made, the order's data is signed with the private key of the sender. Then, the wallet sends the order along with its signature to one or more nodes (relays) in the network.

3. **Relay broadcast**: On the reception of the order, relays update their public order book and broadcast the order to other relays/ring miners. The protocol doesn't require order books to be built in a certain way, such as first-come-first-serve. Instead, relays have the power to make their own design decisions in building their order books.

4. **Ring-mining (order matching)**: Ring Miners receive the order and add it to their order book. Each miner tries to fill it fully or partially at the given exchange rate or better by ring-matching it with multiple other orders. Ring-matching is the main reason why the protocol is able to provide high liquidity over any pair. If the executed rate is better than what the user asked for, the savings (margin) are shared amongst all orders in the ring. As a reward (fee), the miner can choose between claiming the Margin Split and giving back the LRx to the user, or just keeping the LRx fee.

5. **Verification & Settlement**: The ring is received by the Loopring Smart Contract. It makes multiple checks to verify the miner's supplied data and determines if the ring can be settled fully or partially (depending on the fill rate of orders in the ring and the tokens in the users' wallets). If all checks are successful, the contract automatically makes the token transfers to the users and pays the miner's fees at the same time. If the sender's balance as determined by the LSC are insufficient, it will be considered scaled-down. A scaled-down order is not the same as a cancelled order: a scaled-down order will automatically scale up to its original size if sufficient funds are deposited to its address, while cancellation is a one way manual operation and can't be reversed.

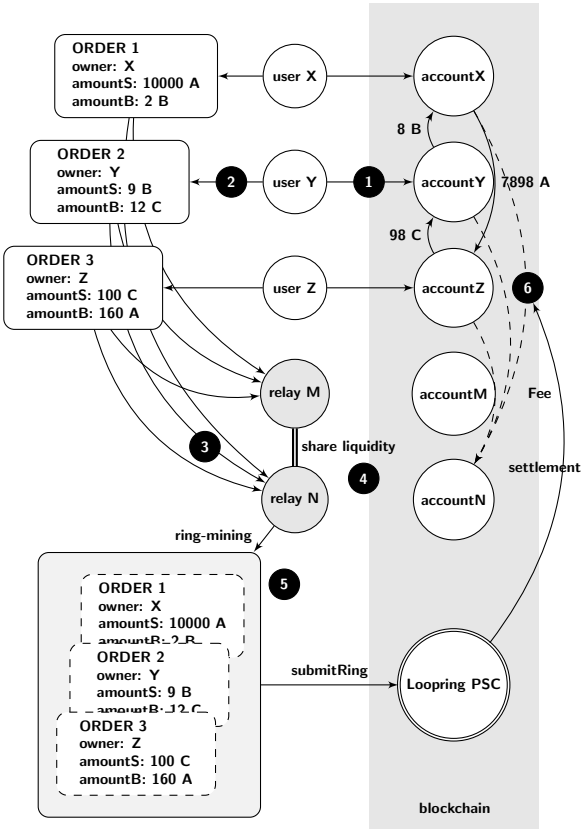

Figure 2: Loopring Exchange Process

6 Business Model Flexibility

It's important to note that Loopring's open standard allows participants significant flexibility in how they conduct business. Actors with novel business models are free to implement them and provide value for their users, earning LRx fees on volume or other metrics in the process, if they so choose. The ecosystem is modular and meant to support participation from a multitude of applications. The non-proprietary protocol places control in the hands of users and collaborators.

6.1 Order Book

Relays can design their order books in any number of ways to display and match users' trades. A first implementation of our own order book follows an OTC model, where limit orders are ranked or positioned based on price alone. Timestamps of orders, in other words, have no bearing on the order book. However, a relay is free to design their

order book in such a way as to emulate a typical centralized exchange's matching engine, where orders are ranked by price, but with a timestamp respecting filter as well. If a relay was inclined to offer this type of order book, they can own/integrate with a wallet, and have those wallet orders sent solely to the single relay, who would then be able to match orders based on time. Any such configuration is possible.

6.2 Liquidity Sharing

Relays are similarly free to design how they share liquidity and orders with each other. Our consortium blockchain is but one solution to accomplish this, and the ecosystem is free to network and communicate as they wish. Besides joining a consortium blockchain, they can build and manage their own, creating rules/incentives as they see fit. Relays can also work solo, as seen in the time-sensitive wallet implementation above. Of course, there are clear advantages in communicating with other relays in pursuit of network effects, however, different business models could merit peculiar sharing designs.

7 Protocol Specification

7.1 Anatomy of an Order

An order is a pack of data that describes the intent of the user's trade. A Loopring order is defined using the Uni-Directional Order Model, or UDOM, as follows:

```
message Order {
  address protocol;
  address owner;
  address tokenS;
  address tokenB;
  uint256 amountS;
  uint256 amountB;
  unit256 lrcFee
  unit256 validSince; // Seconds since epoch
  unit256 validUntil; // Seconds since epoch
  uint8   marginSplitPercentage;  // [1-100]
  bool    buyNoMoreThanAmountB;
  uint256 walletId;
  // Dual-Authoring address
  address authAddr;
  // v, r, s are parts of the signature
  uint8   v;
  bytes32 r;
  bytes32 s;
  // Dual-Authoring private key,
  // not used for calculating order's hash.
  string  authKey;
}
```

To ensure the origin of the order, it is signed against the hash of its parameters, excluding $\_authAddr$, with the user's private key. The $\_authAddr$ parameter is used for signing order rings that this order is part of, which prevents front-running. Please reference 9.1 for more details. The signature is represented by the v, r, and s fields, and is sent alongside the order parameters over the network. This guarantees the order stays immutable during its whole lifetime. Even though the order never changes, the protocol can still compute its current state based on the balance of its address along with other variables.

UDOM doesn't include a price (which must be a floating-point number by nature), but, instead uses the term $rate$, which is expressed as $\frac{amountS}{amountB}$. The rate is not a floating-point number but an expression that will only be evaluated with other unsigned integers on demand, to keep all intermediate results as unsigned integers and increase calculation accuracy.

7.1.1 Buy Amounts

When a miner ring-matches orders, it's possible that a better rate will be executable, allowing you to get more $tokenB$ than the $amountB$ you specified. However, if $buyNoMoreThanAmountB$ is set to $true$, the protocol ensures you receive exactly $amountB$ of $tokenB$. Thus, UDOM's $buyNoMoreThanTokenB$ parameter determines when an order is considered completely filled. $buyNoMoreThanTokenB$ applies a cap on either $amountS$ or $amountB$, and allows users to express more granular trade intentions than traditional buy/sell orders.

Example: with amountS = 10 and amountB = 2, r = 10/2 = 5. Thus the user is willing to sell 5 tokenS for each tokenB. The miner ring-matches and finds the user a rate of 4, allowing the user to receive 2.5 tokensB instead of 2. However, if the user only wants 2 tokensB and set the buyNoMoreThanAmountB flag to true, the LSC performs the transaction at a rate of 4 and the user sells 4 tokenS for each tokenB, effectively saving 2 tokenS. Keep in mind this does not take into account miners' fees (See 8.1).

If we use

```
Order(amountS,tokenS,
      amountB,tokenB,
      buyNoMoreThanTokenB)
```

to represent a order in a simplified form, then for ETH/USD markets on a traditional exchange, traditional buy-sell modeling can express the 1st and the 3rd order below, but not the other two:

1. User wants to sell 10 ETH at price 300 USD/ETH. This order can expressed as $Order(10, ETH, 3000, USD, false)$.

2. User wants to sell ETH at price 300 USD/ETH to get 3000 USD. This order can expressed as $Order(10, ETH, 3000, USD, true)$.

3. User wants to buy 10 ETH at price 300 USD/ETH, This order can expressed as $Order(3000, USD, 10, ETH, true)$.

4. User wants to spend 3000 USD to buy as many ETH as possible at price 300 USD/ETH, This order can expressed as $Order(3000, USD, 10, ETH, false)$.

7.2 Ring Verification

LSC does not perform exchange rate or amount calculations, but must receive and verify what the miner supplies for these values. This is done by miners for two main reasons: solidity does not have support for floating point math, especially pow(x, 1/n), and it is desirable for the computation to be made off-chain to save gas.

7.2.1 Sub-Ring Checking

This step prevents arbitrageurs from unfairly realizing all the margin in a ring by implementing new orders within it. Essentially, once a valid ring is found by a miner, it could be tempting to add other orders to the ring to fully absorb the users' margin (rate discounts). This is zero-risk, zero-value add to the network, and is considered unfair conduct by the miner. To prevent this, Loopring requires that a valid loop cannot contain a sub-ring. To check this, the LSC ensures a token cannot be in a buy or sell position twice. In the below diagram, we can see that tokenA is a sell token twice and a buy token twice, which would be disallowed.

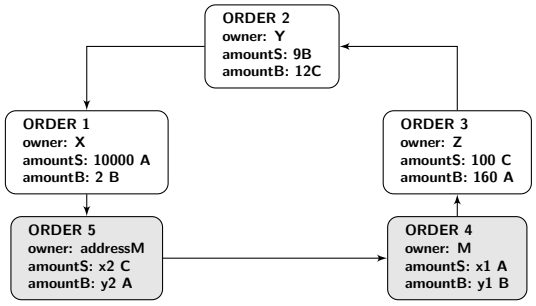

Figure 3: A Ring with Sub-Ring

7.2.2 Fill Rate Checking

The exchange rate calculations in the ring are made by miners for reasons stated above. It is the LSC that must verify they're correct. First, it verifies that the buy rate the miner can execute for each order is at least equal to or less than the original buy rate set by the user. This ensures the user gets at least the exchange rate they asked for or better on the transaction. Once the exchange rates are confirmed, the LSC ensures that each order in the ring shares the same rate (margin) discount. For instance, if the discounted rate is γ, then the price for each order will be:

$r_{0\to1}\cdot(1-\gamma)$, $r_{1\to2}\cdot(1-\gamma)$, $r_{2\to0}\cdot(1-\gamma)$, and satisfied:

$$r_{0\to1}\cdot(1-\gamma)\cdot r_{1\to2}\cdot(1-\gamma)\cdot r_{2\to0}\cdot(1-\gamma)=1 \qquad (2)$$

We can find out:

$$\gamma = 1-\frac{1}{\sqrt[3]{r_{0\to1}\cdot r_{1\to2}\cdot r_{2\to0}}}.$$

In the other circumstance, if transaction cross n orders, the `discount` is:

$$\gamma = 1-\frac{1}{\sqrt[n]{\prod_{i=0}^{n-1} r^i}},$$

where r^i is the order turnover rate of i-th order. Obviously, only when the discount rate is $\gamma \geq 0$, these orders can be filled; and the i-th order's O^i actual exchange rate $\hat{r^i} = r^i\cdot(1-\gamma)$, $\hat{r^i} \leq r^i$.

7.2.3 Fill Tracking & Cancellation

A user can partially or fully cancel an order by sending a special transaction to the LSC, containing the details about the order and the amounts to cancel. The LSC takes that into account, stores the amounts to cancel, and emits an OrderCancelled event to the network. The LSC keeps track of fill and cancellation amounts by storing their values using the order's hash as an identifier. This data is publicly accessible and OrderCancelled / OrderFilled events are emitted when it changes. Tracking these values is critical for the LSC during the ring settlement step.

7.2.4 Order Scaling

Orders are scaled according to the history of filled and cancelled amounts and the current balance of the senders' accounts. The process finds the order with the smallest amount to be filled according to the above characteristics and uses it as a reference for scaling all transactions in the ring.

Finding the lowest value order can help to figure out the fill volume for each order. For instance, if the i-th order is the lowest value order, then the number of tokens sold from each order $\hat{s}$ and number of tokens purchased $\hat{b}$ from each order can be calculated as:

$$\begin{aligned}
&\hat{s}^i = \overline{s}_i,\ \hat{b}^i = \hat{s}^i/\hat{r}^i,\ ;\\
&\hat{s}^{i\oplus1} = \hat{b}^i,\ \hat{b}^{i\oplus1} = \hat{s}^{i\oplus1}/\hat{r}^{i\oplus1};\\
&\hat{s}^{i\oplus2} = \hat{b}^{i\oplus1},\ \hat{b}^{i\oplus2} = \hat{s}^{i\oplus2}/\hat{r}^{i\oplus2};\\
&...
\end{aligned}$$

where $\overline{s}_i$ is the the balance left after orders are partially filled.

During implementation we can safely assume any order in the ring to have the lowest value, then iterate through the ring at most twice to calculate each orders' fill volume.

Example: If the smallest amount to be filled compared to the original order is 5%, all the transactions in the ring are scaled down to 5%. Once the transactions are completed, the order that was considered to have the smallest amount remaining to be filled should be completely filled.

7.3 Ring Settlement

If the Order Ring fulfills all the previous checks, the ring can be closed, and transactions can be made. This means that all the n orders O form a closed ring of orders, connected as in the figure below:

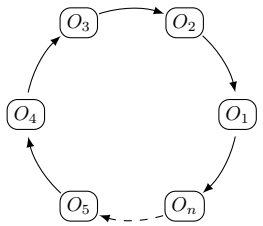

Figure 4: Ring Settlement

To make the transactions, the LSC uses the $TokenTransferDelegate$ smart contract. The introduction of such a delegate makes upgrading the protocol smart contract easier as all orders only need to authorize this delegate instead of different versions of the protocol.

For each order in the ring, a payment of $tokenS$ is made to the following order. Then the miner's fee is paid depending on the fee model chosen by the miner. An $OrderFilled$ event is then emitted. Finally, once all the transactions are made, a $RingMined$ event is emitted.

7.3.1 Emitted Events

The protocol emits events that allow relays, order browsers, and other actors to receive order book updates as efficiently as possible. The emitted events are:

- **OrderCancelled**: a specific order has been cancelled.
- **OrdersCancelled**: all orders of a trading pair from an owning address have been cancelled.
- **AllOrdersCancelled**: all orders of all trading pairs from an owning address have been cancelled.
- **RingMined**: A ring has been settled successfully. This event contains data related to each inner-ring token transfer.

8 LRx Token

LRx is our generalized token notation. LRC is the Loopring token on Ethereum, LRQ on Qtum, and LRN on NEO. Other LRx types will be introduced in the future as Loopring is deployed on other public blockchains.

8.1 Fee Model

When a user creates an order, they specify an amount of LRx to be paid to the miner as a fee, in conjunction with a percentage of the margin made on the order that the miner can claim. This is called the margin split. The decision of which one to choose is left to the miner.

A representation of the margin split:

If the margin on the ring is too small, a miner will choose the LRx fee. If, on the contrary, the margin is substantial enough for the resulting margin split to be worth more than the LRx fee, a miner will choose the margin split. There is another proviso, however: when the miner chooses the margin split, they must pay the user (order creator) a fee, which is equal to the LRx the user would have paid to the miner as a fee. This increases the threshold of where the miner will choose the margin split to twice the LRx fee of the order, increasing the propensity of the LRx fee choice. This allows miners to receive a constant income on low margin rings for the tradeoff of receiving less income on higher margin rings. Our fee model is based on the expectation that as the market grows and matures, there will be fewer high margin rings, thus necessitating fixed LRx fees as incentive.

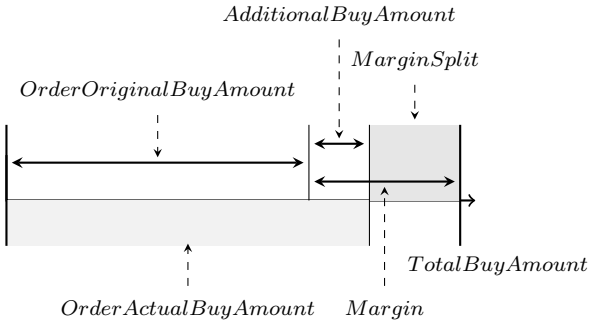

Figure 5: A 60% Margin Split

We end up with the following graph:

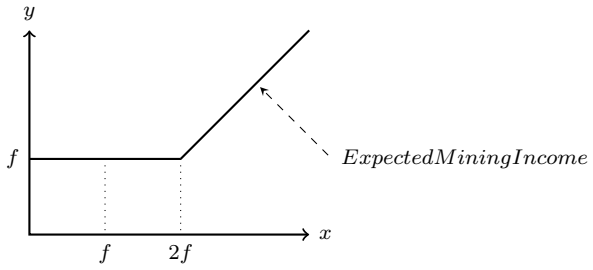

Figure 6: Loopring Fee Model

where f is the LRx fee, x is the margin split, y is the mining income. $y = max(f, x - f)$ as indicated by the solid line; if the LRx fee for the order is 0, the equation is $y = max(0, x - 0)$ that simplifies to $y = x$ as indicated by the dashed line.

The consequences are:

1. If the margin split is 0, the miners will choose the flat LRx fee and are still incentivized.
2. If the LRx fee is 0, the orange line results and the income is based on a general linear model.

3. When the margin split income is greater than 2x(LRx fee), the miner chooses the margin split.

It should be noted that if the LRx fee is non-zero, no matter which option the miner chooses, there will always be a transfer of LRx between the miner and the order's sender. Either the miner earns the LRx fee, or pays the LRx fee back to the sender to take the margin split.

Ring miners will share a certain percentage of fees with wallets. When a user places an order through a wallet and it is filled, the wallet is rewarded with a portion of the fees or margin split. Wallets represent a primary target for protocol integration as they have the user base, but little source of income.

8.2 Decentralized Governance

LRx tokens are used to align the financial incentives of the various participants in the network. Such alignment is necessary for broad adoption of the protocol, whose success rests largely on improving liquidity in a robust decentralized ecosystem.

LRx tokens will be used to effectuate protocol updates through decentralized governance. Smart contract upgrades will be voted on by token holders to dissuade rogue forks from potentially siphoning off liquidity through incompatibility. Upgradeability is crucial to the protocol's success as it must adapt to market demands, and the changing underlying blockchains.

9 Fraud and Attack Protections

9.1 Front-running Prevention

In decentralized exchanges, front-running is when someone tries to copy another node's ring solution, and have it mined before the original transaction that is in the pending transaction pool (mempool). This can be achieved by specifying a higher transaction fee (gas price). The major scheme of front-running in Loopring (and any protocol for order-matching) are order-filch: when a front-runner steals one or more orders from a pending ring settlement transaction; and Ring-filch: when a front-runner steals the entire ring from the pending transaction.

When a submitRing transaction is not confirmed and still in the pending transaction pool, anyone can easily spot such a transaction and replace *miner_address* with their own address (the filcher_address) , then they can re-sign the payload with filcher_address to replace the ring's signature. The fincher can set a higher gas price and submit a new transaction hoping block miners will pick his new transaction into the next block instead of the original submitRing transaction.

Previous solutions to this problem had important downsides (v1.1): requiring more transactions and thus cost miners more gas; and taking at least twice the blocks to settle a ring. Our new solution involves the mechanism of setting up two levels of authorization for orders - one for settlement, and one for mining.

How it works:

1. For each order, the wallet software will generate a random public-key/private-key pair, and put the key pair into the order's JSON snippet. (An alternative is to use the address derived from the public-key instead of the public-key itself to reduce byte size. We use $authAddr$ to represent such an address, and $authKey$ to represent $authAddr$'s matching private-key).

2. All fields in the order except $authKey$ is signed using the *owner* address's private-key (not $authKey$) as shown in the image below.

3. The wallet will send the order, together with the $authKey$ to miners (relays) for matching.The miner will verify that $authKey$ and $authAddr$ are correctly paired and the order's signature is valid with respect to owner_address.

4. When a ring is identified, the miner will use each order's $authKey$ to sign the ring's hash, *miner_address*, and all the mining parameters. In the example below, the ring contains 3 orders, therefore there will be 3 signatures by the 3 $authKey$s. We call these signatures the *auth_signature*s. The miner also needs to sign the ring's hash together with all mining parameters using *miner_address*'s private-key.

5. The miner calls the submitRing function with all the parameters, as well as the 3 extra *auth_signature*s. Notice that $authKey$s are NOT part of the on-chain transaction and thus remain unknown to people other than the relay itself, as shown in the image below.

6. The Loopring Protocol will now verify each *auth_signature* against the corresponding $authAddr$ of each order, and reject the ring if any *auth_signature* is invalid.

[TODO: daniel wang]

The result is that now:

- The orders signature (by the private-key of the *owner_address*) guarantees the order cannot be modified, including the $authAddr$.
- The miners signature (by the private-key of the *miner_address*) guarantees nobody can use his identity to mine a ring.
- The *auth_signature*s guarantees the entire ring cannot be modified, including *miner_address*. And since ring-filchers do not have access to $authKey$s, they cannot re-generate a new set of *auth_signature*s thus are unable to generate a filch transaction.

Dual Authoring prevents ring-filch and order-filch while still ensuring the settlement of rings can be done in one single transaction. In addition, Dual Authoring opens doors for relays to share orders in two ways: non-matchable sharing and matchable sharing. Loopring operates an OTC model and only supports limit-price orders, meaning that orders timestamps are totally ignored. This implies that front-running a trade has no impact on the actual price of that trade, but does impact whether it gets filled or not.

10 Other Attacks

10.1 Sybil or DOS Attack

Malicious users - acting as themselves or forged identities - could send a large amount of small orders to attack Loopring nodes. However, since we allow nodes to reject orders based on their own criteria - which they may hide or reveal - most of these orders will be rejected bfor not yielding satisfying profit when matched. By empowering relayers to dictate how they manage orders, we do not see a massive tiny order attack as a form of unethical behaviour.

10.2 Insufficient Balance

Malicious users could sign and spread orders whose value inside the order is non-zero but whose address actually has zero balance. Nodes could monitor and notice that some orders actual balance is zero, update these order states accordingly and then discard them. Nodes must spend time to update the status of an order, but can also choose to minimize the effort by, for example, blacklisting addresses and dropping related orders.

11 Summary

The Loopring protocol sets out to be a foundational layer for decentralized exchange. In so doing, it has profound repurcussions in how people exchange assets and value. Money, as an intermediate commodity, facilitates or replaces barter exchange and solves the double coincidence of wants problem [18], whereby two counterparties must desire each other's distinct good or service. Similarly, Loopring protocol aims to dispense of our dependencies on coincidence of wants in trading pairs by using ring matching to more easily consummate trades. This is meaningful for how society and capital markets transfer value with tokens, traditional assets, and beyond. In addition, with Dual Authoring, Loopring solves the real and pernicious problem of front running faced by all decentralized exchanges and their users today.

- Off-chain order management and on-chain settlement = no sacrifice in performance for security. Network is maintained by a self-motivated group of relays who have flexibility in running their order books.
- Network is maintained by a self-motivated group of relays who have flexibility in running their order books and communicating.
- Greater liquidity due to more counterparties available through order sharing, and higher probability that any counterparty can be a useful trading partner due to multi-party trades.
- Orders can be propagated through arbitrary communication mediums, and liquidity siloes can be connected.
- Free, public smart contracts enable any dApp to build or interact with the protocol.
- Standardization among operators allows for network effects, deeper liquidity and an improved end user experience.
- Reduced barriers to entry for market-makers = lower costs for nodes joining the network and end users.
- Anonymous trading directly from users wallets.

12 Acknowledgements

We would like to express our gratitude to our mentors, advisers and to the many people in the community that have been so welcoming and generous with their knowledge. In particular, we would like to thank Shuo Bai (from ChinaLedger); Professor Haibin Kan; Alex Cheng, Hongfei Da; Yin Cao; Xiaochuan Wu; Zhen Wang, Wei Yu, Nian Duan, Jun Xiao, Jiang Qian, Jiangxu Xiang, Yipeng Guo, Dahai Li, Kelvin Long, Huaxia Xia, Jun Ma, and Encephalo Path for reviewing and providing feedback on this project. We also welcome more feedback from the community.

References

[1] Vitalik Buterin. Ethereum: a next generation smart contract and decentralized application platform (2013). *URL {http://ethereum. org/ethereum. html}*, 2017.

[2] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[3] Patrick Dai, Neil Mahi, Jordan Earls, and Alex Norta. Smart-contract value-transfer protocols on a distributed mobile application platform. *URL: https://qtum. org/uploads/files/cf6d69348ca50dd985b60425ccf282f3. pdf*, 2017.

[4] Viktor Atterlönn. A distributed ledger for gamification of pro-bono time, 2018.

[5] Hernando de Soto. *The Mystery Of Capital*. Basic Books, 2000.

[6] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[7] Fabian Schuh and Daniel Larimer. Bitshares 2.0: Financial smart contract platform, 2015.

[8] Bancor protocol. *URL https://bancor.network/*, 2017.

[9] Yaron Velner Loi Luu. Kybernetwork: A trustless decentralized exchange and payment service. `https://kr.kyber.network/assets/KyberNetworkWhitepaper.pdf`, Accessed: 2018-03-05.

[10] Will Warren and Amir Bandeali. 0x: An open protocol for decentralized exchange on the ethereum blockchain, 2017.

[11] Wikipedia. Coincheck. `https://en.wikipedia.org/wiki/Coincheck`, Accessed: 2018-03-05.

[12] Wikipedia. Mt. gox. `https://en.wikipedia.org/wiki/Mt._Gox`, Accessed: 2018-03-05.

[13] Robert McMillan. The inside story of mt. gox, bitcoins 460 dollar million disaster. 2014.

[14] Rossella Agliardi and Ramazan Genay. Hedging through a limit order book with varying liquidity. 2014.

[15] Iddo Bentov and Lorenz Breidenbach. The cost of decentralization. `http://hackingdistributed.com/2017/08/13/cost-of-decent/`, Accessed: 2018-03-05.

[16] Daniel Wang. Coinport's implemenation of udom. `https://github.com/dong77/backcore/blob/master/coinex/coinex-backend/src/main/scala/com/coinport/coinex/markets/MarketManager.scala`, Accessed: 2018-03-05.

[17] Fabian Vogelsteller. Erc: Token standard. *URL https://github.com /ethereum /EIPs /issues /20*, 2015.

[18] Nick Szabo. Menger on money: right and wrong. `http://unenumerated.blogspot.ca/2006/06/menger-on-money-right-and-wrong.html`, Accessed: 2018-03-05.

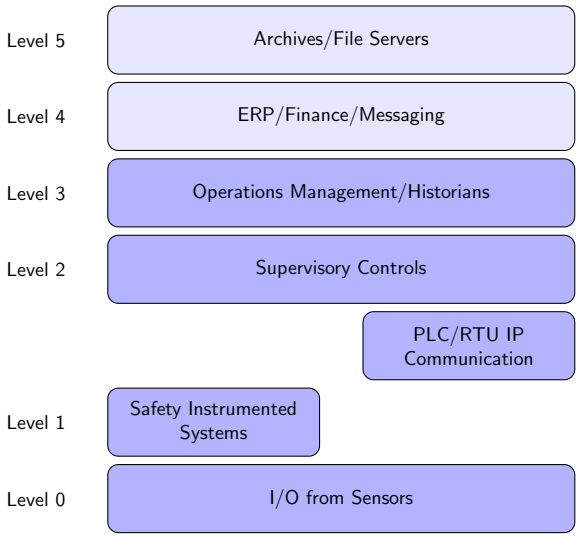

Figure 7: FOSS in Chrome influences industry structure by increasing competition

Appendices

Appendix A Protocol Ethereum Smart-Contracts

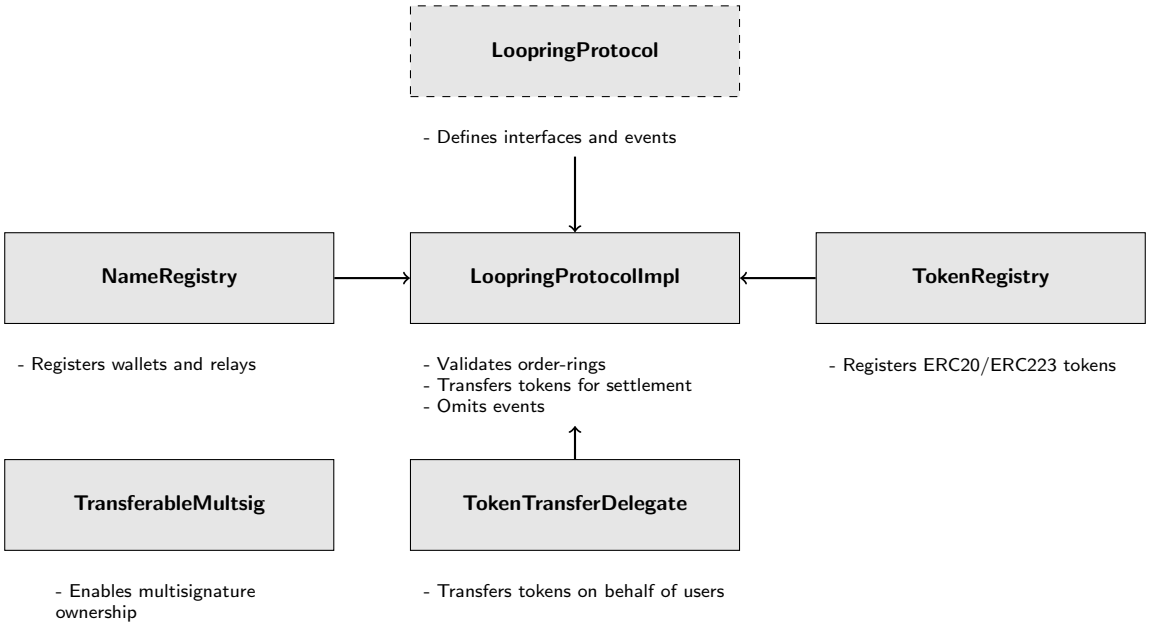

Figure 8: Loopring Ethereum Smart-Contracts

The source code of these smart contract are available at:
https://github.com/Loopring/protocol/tree/master/contracts.