Contents

1 Dark automated match engine.

Non-custodial order book over darkfi blokchcain. In this model the traders doesn't have to interact after the swap for transaction signature, instead settlement node which owns the coins, can transfer the swapped tokens directly, since the first part of the swap is already done in the exchange transaction by burning the base coins.

1.1 order type

the match order only accept limit orders.

1.2 exchange transaction

Assume a trader owns a coin c of Token t_1 in order to add a an order in the order book with public key $pk^{matcher}$, the trader burns c, mint new $c' = (pk_1^{matcher}, v_1, t_1, spendhook_1, userdata_1, coinblind_1, tokenblind_1)$, create an order with quote token id t_2 commit to it with $tokenblind_2$, quote rate p (quote price per base price), timeout duration, τ, constrain order commit for order o = (t_2, p, τ), the proof need to commit to withdraw public key $pk^{withdraw}$ that can be used in case of withdraw or settlement. note, no need to reveal $pk^{matcher}$ in the public inputs to restrict the swap to specific exchange since the coin is minted with that matcher pk

1.2.1 send the gas fee in the order details (TODO)

exchange proof need to commit to gas fee value, gas fee token.

1.3 match proof

It's a proof of knowledge that the match engines have access two coins with non-negative spread (this is two steps, the coin values are sufficient, and the rate matches), two coins are of the opposite direction, the counter party token is the desired token. for a match between two parties left (l), and right (r) prove konwledge to two coins l, r, such that:

base token of the left token is the same as quote of the right token, quote of the left token is the same as base of the right token, the exchange rate of both parties has non-negative spread.

- $commit(t_1^l, tokenblind_1^l) == commit(t_2^r, tokenblind_2^r)$
- $commit(t_2^l, tokenblind_2^l) == commit(t_1^r, tokenblind_1^r)$

the following three rules make sure that the two parties has non-negative spread.

- $p_1 p_2 >= 1$
- $v_2^r >= v_1^l$
- $v_2^l >= v_1^r$

constrain, and reveal the two coins used.

1.3.1 spread difference (TODO)

the spread should be added to the book, commit to the spread difference value, it will be spent along with the fee transaction upon swap success.

1.4 order settlement transaction

settlement node, would combine match proof with transfer proofs in a single transaction.

1.5 withdraw transaction

trader who issued a exchange transaction can withdraw, and spend the spend hook minted token as long as the settlement tx isn't broadcasted yet, or if the order timeout duration is passed.

1.5.1 exchange transaction spend hook

if the duration τ specified in the order is timeout, the spend hook is executed, and the a coin of the same value is minted.

1.6 cancel order

at any moment as long as the order isn't settled, the trader can cancel the order by minting a new coin, or if the order duration is timedout, spend hook contract can mint new Coin of the same Token and value back to the Trader owned by $pk^{withdraw}$

1.6.1 cancel order before timeout

the match engine need to be notified by cancel event, for this, the burn proof in the previous exchange transaction is combined with mint proof to create new transfer proof. it seems that the burn proof will be reused, and the nullifier will be issued twice, but in fact the exchange transaction will not be finalized until it's settled, with final swap transfer, and match engine fee contract executed, until then the burn proof in exchange transaction shouldn't be finalized, and it's finalized shouldn't be added to spent coins.

1.6.2 on timeout

the exchange transaction is finalized, match engine ought to remove bid/ask orders, withdraw spend hook should be executed.

how can match engine remove the order if the trader cancels the order?

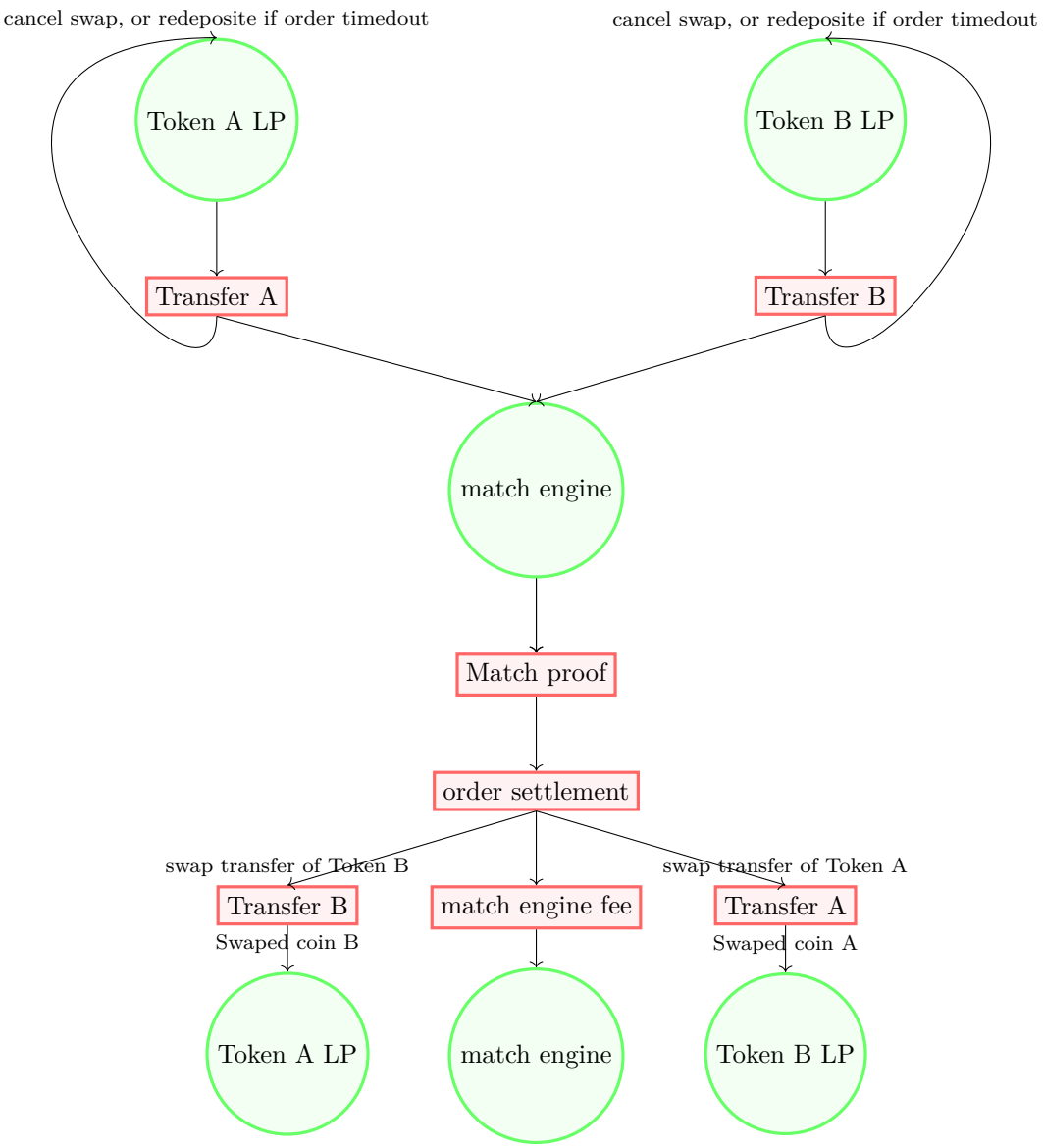

Figure 1: match engine contracts