Decentralized Cross Chain Token Exchange Based on Atomic Swap

Li Shiwei

Introduction:

Decentralized token exchange protocol has drawn a lot of attentions in cryptocurrency world recently. Protocols like 0x and kyber has already shown a good example to the world what is a decentralized exchange. The idea has already been approved by the market.

The basic assumption and principle for the decentralized exchange protocol like 0x and kyber is to use the underlying blockchain as the final balance settlement for both traders. This assumption certainly set a hard limitation for the protocol. Its performance and cost of is bounded by the underlying blockchain. Take ethereum for example, currently ethereum can only handle 10-30 transactions per second. Protocol like 0x and kyber will need at least 1 transaction for every exchange activity.

As a result, based on this assumption, the decentralized exchange cannot handle exchange in large volume. Considering one of the biggest centralized exchange Binance as a example, in peak hour, Binance can handle hundreds of thousands transaction every second. This will be an unachievable task for the decentralized exchange. Even the whole ethereum network works only for the decentralized exchange. In maximum it can only handle 10-30 transaction per second. That is far below users' expectation in the market.

In this paper we aims to describe a decentralized exchange system that does not use native blockchain as the underlying balance settlement for every exchange activity. Instead, we use another private chain or public chain with a higher transaction speed capacity for users balance settlement in a user defined period of time. The main blockchain, will sync the balance when user opts to withdrawn from the "high speed" exchange platform.

Decentralized exchange:

Before go depth into the system design, we would like to first clarify the definition of the decentralized exchange, and its main attributes which different itself from the centralized exchange.

For centralized exchange, users are expected to deposit their cryptocurrency to the exchange wallet address before trading. Technically, the centralized exchange is the owner of the cryptocurrency that is deposited, it can freely transfer the cryptocurrency to other addresses, limit users from withdraw their cryptocurrency to their own wallet address, even declare bankrupt and run away with all the cryptocurrency under its control.

For centralized exchange, the exchange is responsible to fund's security, users need to trust the exchange will act honestly and securely.

Decentralized exchange in another way, user are not expected to fully trust any single party in the exchange workflow.

1. Users are not expect to deposit their token to a wallet address which is not under user's control. No party can move user's fund without user's concern.
2. The fund security during the exchange process is not guaranteed by any single party. The underlying blockchain and all the parties participated in the blockchain provide the infrastructure to honestly execute the trading logic and users themself are responsible for their fund security, i.e. they should not lost their private key to control the fund.

Decentralized exchange protocol like 0x and kyber network achieved these attribute by using the underlying blockchain(ethereum) as the computation platform to honestly execute trading logic. However, as stated above, the overall performance also heavily affected by the blockchain platform it uses.

Here we would like to disucss using a private blockchain with higher speed capacity as the computation platform for trading logics, at the same time give enough incentive for other 3rd parties to join in the private blockchain as an "auditor" to increase the credibility of the trading platform.

Exchange workflow:

In order to simplify this paper, we just use ethereum in private network with a higher performance consensus algorithm (POA). It is free to use another blockchain as long as it provide basic scripting functionality.

Consider a case, user A have token X in the ethereum mainnet and would like to trade for token Y. A can go through the following workflow:

1. User A "deposit" token X in ethereum mainnet to a wallet address
2. User A gets a 1:1 amount of token X on the private chain.
3. User A trade with other users on this exchange platform to get token Y
4. User A "deposit" token Y in private chain to a wallet address
5. User A get a 1:1 amount of token Y on ethereum mainnet.

Take note that even we use the word deposit in step 1 and step 4. As a decentralized exchange solution, we need to ensure users do not need to trust any single part in the process and the security of their fund is guaranteed by the technology itself.

In order to achieve this goal 3 major problems need to be resolved.

1. The deposit process need to be "atomic", in other word, the process can only be success or fail. If either party find the counterpart play dishonestly, it should be able to withdraw the operation without any loss.
2. The 1:1 token exchange between 2 seperate blockchain need to be available all the time. The exchange channel should not under the control of a single party.

3. The computation in step 3 need to be audited by other parties. And the audit party should have enough incentive to participate in the auditing activity.

Atomic token swap:
The deposit process can use “atomic swap” to get a 1:1 exchange of token in mainnet and private chain.

The concept of atomic swap is not new, it is widely adopted in lightning network, raiden network and etc. Technically it is achieved by deployed a hash time-locked contracts(HTLC) on the mainnet and the private chain. Consider a case Alice would like to give 100 X token to Bob on mainnet in exchange for 100 Y token on private net.

1. Alice generate a random secret key and hash it to generate a hash lock.
2. Alice use the hash lock to create a swap contract to send Bob 100 X token on mainnet.
3. After seen Alice’s swap contract on mainnet, Bob also create a swap contract to send Alice 100 Y token on private blockchain with the hashlock used by Alice in the mainnet.
4. After seen Bob’s swap contract on private blockchain, Alice checks the contract and verify the details. Then Alice can use the secret key to retrieve the 100 Y token on private blockchain from Bob’s swap contract.
5. After Alice reveal her secret key. Bob checks the secret key on private blockchain. Then Submit the same secret key on mainnet to retrieve the 100 X token on mainnet.

In the above use case, if Alice does not reveal her secret key in step 4. Alice and Bob can both wait until a certain amount of time defined in the time hash lock contract. Then request to refund.

Atomic swap is a ready solution to trustlessly transfer fund cross blockchain. The only problem left is how to guarantee the X token in the mainnet have the same representation of its value of Y token on the private blockchain ?

This can be easily done by only allowing the owner of token X on mainnet to lock up a certain amount of token X. Then issue a new token Y with same amount of token X locked up on the mainnet. The token Y creator need to show evidence that he is also the owner of token X.

Here we propose an enhancement of ERC20 token standard to achieve above feature.

The added interfaces are:
/* token owner need to generate a random secret to create token on mainnet and private blockchain. The same secret hash is shown as a evidence that the token are generated from the same owner, for the same project, and represent the same usage and intrinsic value */
bytes32 public ownerSignatureHash; //hash value of owner's secret and token name.

// create a hashlock, used in step 2 and 3 as the example above, returns the contractId to index the contract.

```
function open (address _receiver, bytes32 _hashlock, uint _timelock, uint _amount) public returns (bytes32 contractId)

// reveal secret to the hashlock to retrieve token, used in step 4 and 5 as the example above
function close(bytes32 _contractId, bytes32 _secret) public returns (bool)

// in case of contract expires, use the method to refund from the hashlock
function refund(bytes32 _contractId) public returns(bool)

// use the method to check the hashlock details and secret,  used in step 4 as the example above
function checkContract(bytes32 _contractId) public view returns (
    address sender,
    address receiver,
    uint amount,
    bytes32 hashlock,
    uint timelock,
    bytes32 secret
  )
```

These method give ERC20 token the ability to trade cross-chain and also give unique identity to identify the same token cross-chain.

Token listing setup:
Consider Alice as the project owner of token X and hold 1000 tokens. She would like to list the token X in this private blockchain exchange platform.

We need to ensure that as the project owner of token X, Alice can create 1000 Y tokens on the private chain to give initial “liquility” to the private chain market. At the same time, we need have a mechanism to ensure Alice can always provide a 1:1 exchange between X and Y, i.e. She should not have the technical ability to create 1000 Y tokens on the private chain and move her 1000 X tokens on the mainnet to somewhere else, leaving the holder of 1000 Y token with no one to trade back.

In this solution, we propose Alice to deposit her 1000 X token to a multi-signature contract which is owned by platform owner, Alice and auditors if any.

When Alice deposit her 1000 X token into the multi-signature contract, all the involved parties (platform owner, Alice, and auditors) need to multi-sign on the private blockchain to agree to “mint” 1000 Y token to Alice’s account on private blockchain. Then Alice can use atomic token swap mechanism to change tokens Y with other traders from mainnet. Let’s say Bob would like to exchange 500 X token with Alice for token Y. In the end, Alice should have 500 X token on the mainnet, and 500 Y token on the private blockchain. Total tradable number of Y token on

the private blockchain is 1000. The multi-sign wallet on the mainnet also have 1000 X token as reserve.

Suppose Alice now would like to withdraw her 500 Y token from private blockchain to mainnet. She can “burn” her 500 Y token by send 500 Y token back to the Y contract address, Then all the involved parties (platform owner, Alice and auditors) can multi-sign again on the mainnet to withdraw 500 X token from reserve to Alice’s account. At this time, Alice have 1000 X token on the mainnet, Bob have 500 Y token on the private blockchain, Total tradable number of Y token on the private blockchain is 500. The reserved X token on the mainnet is 500. The following table shows the flow of tokens in this example.

Action	Alice X	Alice Y	Bob X	Bob Y	X reserves	Y total supply
Alice deposit 1000 X	0	1000	500	0	1000	1000
Alice exchange 500 Y with Bob	500	500	0	500	1000	1000
Alice withdraw 500 Y	1000	0	0	500	500	500

From the above example, we can realize that:

1. The X reserves are always equal to Y’s total supply on the private blockchain
2. Withdraw from private blockchain will cut the total supply of token Y.
3. Trader can always choose to trade with other trader from mainnet to get the same result as withdraw.

Alice’s first deposit can be seen to create initial “liquility” of Y token on the private blockchain for other trade activity to start with. The platform use multi-sign mechanism to lock the X token in the reserve, as a result, this guarantee that for every Y token on the private blockchain. User can always exchange it back to X token on the mainnet.

In fact, it is always more efficient to exchange with other traders to “deposit” or “withdraw” to the private blockchain. If you choose to “burn” Y token by sending back Y token to Y’s contract address, multi-sign withdraw from reserves will have to use at least 2 transactions on the mainnet -- one signature from Alice and one signature from platform owner. That means more gas will be costed, and someone eventually have to pay for this.

Incentive for auditors to joining the private blockchain:

In the above mechanism, the end user, i.e. traders should still should not fully trust this private blockchain trading platform for the following reasons:

1. The private blockchain is controlled only by the trading platform. The trading platform have the technical capability to edit the blockchain data to cheat.
2. The multi-sign mechanism to ensure 1:1 token exchange across blockchain is only controlled by the token project owner and platform owner. If they agree to behave dishonestly, traders can lost all their token.

This brings out the importance of “auditors” to participate in this private blockchain to enhance the credibility of this blockchain as the computation infrastructure. In addition, we should allow other parties to join in the multi-sign reserve pool. However, to become an auditor, there is intrinsic cost like maintain the blockchain node server and responsible to it. There should be enough economic incentive for 3rd parties to participate in.

The obvious incentive for the token project owners are, they would like to list their token to as many trading platforms as possible. As a result, it is reasonable to require token project owner to setup node server to join in the blockchain network. If there are enough number of token listed in the platform, It is unlikely that token project owners and platform owner can all agree to cheat at the same time. Considering EOS has only 21 nodes to support the whole computation infrastructure, this mechanism could bring enough credibility to the market.

We can also give monetary incentive for 3rd party to join in the multi-sign list. One may already realize that, the atomic swap from mainnet to the private blockchain will need at least 1 transaction on the mainnet, in addition there should be someone to generate the secret hash pair for the hashlock. There is intrinsic cost with it and it is reasonable to request user to pay a minor fee for the “deposit” and “withdraw” service.

The fee can be pay in the form of tokens, for example, deposit from mainnet to private blockchain in a ratio of 100: 95. The 5 tokens on the private blockchain are payed to the traders as the deposit fee. Technically, anyone who have token on the private blockchain can act as the trader to provide “deposit” and “withdraw” service. In order to acquire the information of who is asking for “deposit” or “withdraw” service, the trader will have to setup node server for both mainnet and private blockchain. They can also be selected as auditors for the private blockchain. We need to recall that, the purpose to setup private blockchain for trading platform is to achieve a higher speed of decentralized trading compare with the mainnet. So the number of auditors should be limited to a certain amount in order to guarantee the speed.

The decentralized trading logic in the private blockchain can use existing protocol like 0x. We will not discuss further details here.

Conclusion:

This paper provide a mechanism to sync asset from mainnet to a private blockchain for higher speed operation. Although we use exchange platform as an example to elaborate the mechanism. The actual usage of the principles should not only limit to exchange application. Actually, as the private blockchain works as a general computation platform. Any crypto asset on ethereum mainnet can be “deposit” to private blockchain application for high speed operation.

For example, we can run a crypto-kitty like games on the private blockchain for high speed operation and “withdraw” the ERC20 crypto asset (if any) in the game back to the mainnet. In this way, we can take the advantage of the mainnet’s user base and its monetary advantage, at the same time, enjoy the high speed operation of the private blockchain for better user experience.