Migretor: Decentralized cryptocurrency Exchange platform

Team at beecrypt.io

June 18, 2018

1 Abstract

We envision a decentralized cryptocurrency exchange design and a product, where its users should be able to exchange two different blockchain values without involving a trusted third party or centralized entity, keeping in mind privacy, security as paramount of importance with the ease of use with the help of blockchain technologies inline with the spirit of decentralization.

Migretor allows users to store cryptocurrency, transact, and exchange values with other cryptocurrency without involving a trusted third party wholly in a decentralized setup. The decentralized exchange is achieved through Atomic Cross-Blockchain Interoperability.

We have already created a product which is capable of achieving Atomic swaps between Bitcoin and Ethereum blockchain. We are going on to discuss in this document how the Atomic swaps are achieved, the architecture, protocol involved in creating Migretor product in detail further in this document. The atomic swaps can also be extended to other blockchains such as Litecoin, NXT, ARDOR etc.

2 Problem statement

If not all, most of the centralized cryptocurrency exchanges have been hacked, costing its users 100's of millions dollars. The centralized exchanges have no obligations to disclose un-eventful event and continue to operate in factional reserves without transparency which is unfair its users. Centralized exchanges needs trust which is opposite to spirit of decentralized cryptocurrency. Other problems with centralized exchanges is delay in withdrawals, exit scams, privacy issues, manipulations in exchange, seize to operate with intervention of regulatory bodies or changes in regulatory policies, Distributed denial of service(DDoS), systematic withholding of client funds etc.

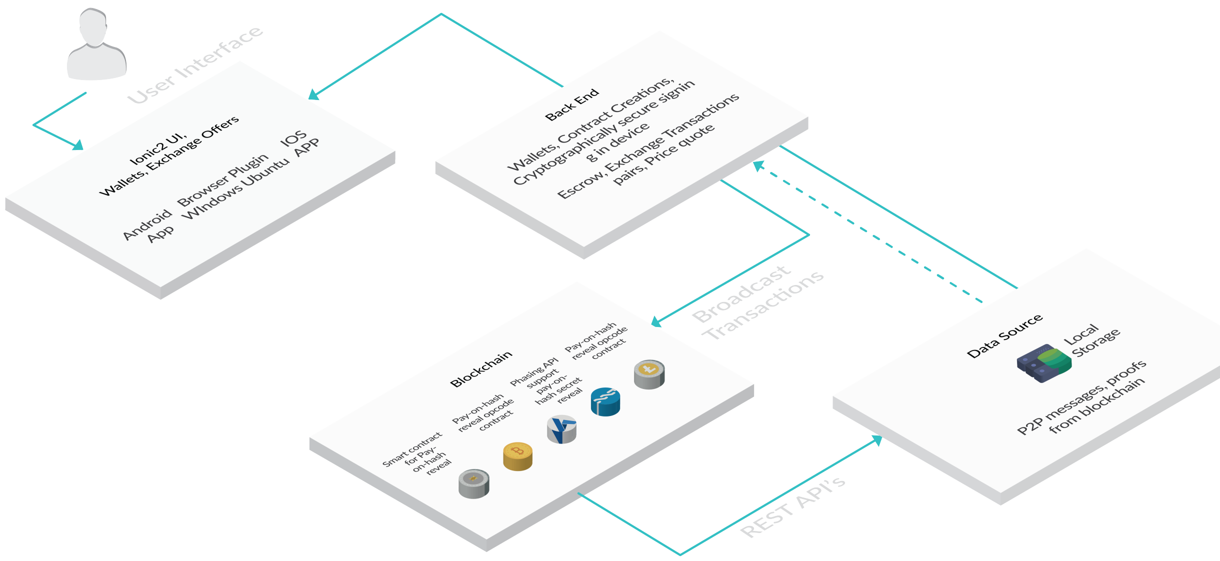
3 Solution

We have created decentralized cryptocurrency exchange, Migretor, where user will have full control of their funds. Following are the features of the Migretor decentralized exchange:

- On chain transactions: Blockchain based Atomic swaps or Atomic cross chain transactions
- Secure: no custodian or No trusted third party for funds, No Arbitrators, no counterparty risk on your funds
- Total control: no limits for deposits and withdrawals of funds, your wallet private never leaves your device, all transactions signed locally before broadcasted, hence offers unprecedented level of security for users funds stored that are stored in exchange.
- Open source: code is available for audit
- Transparent: track your exchange transactions with ease
- Cost effective: Atomic swaps can be done in a fraction of cost compared to fee charged on centralized exchanges
- Robust: effective against DDoS
- p2p and private: quotes, exchange pair are encrypted and only the parties involved in the exchange can view the trades and transactions.
- Any-to-Any exchange pairs

4 Architecture

This section describes over all high-level architecture of the Migretor decentralized cryptocurrency exchange platform including the subsystem-level interaction with Atomic cross chain interportability.


User Interface
Ionic2 UI,
Wallets, Exchange Offers
Android App
Browser Plugin
WIndows Ubuntu
IOS APP
Back End
Wallets, Contract Creations,
Cryptographically secure signin g in device
Escrow, Exchange Transactions pairs, Price quote
Broadcast Transactions
Blockchain
Smart contract for Pay-on-hash reveal
Pay-on-hash reveal opcode contract
Phasing API support pay-on-hash secret reveal
Pay-on-hash reveal opcode contract
Data Source
Local Storage
P2P messages, proofs from blockchain
REST API's

Architecture diagram involving Atomic cross-blockchain interportability and over all decentralized cryptocurrency exchange.

The main architectural goals is, Low coupling well-structured computer system, and combined with high cohesion, supports the general goals of high readability and maintainability. Also, we are keeping in mind to have a modular approach during the software architecture. The architecture subsystems explained in the following sections:

4.1 User Interface

The user interface is implemented using Ionic 2 framework, the framework provides developers to develop mobile based application with ease and apps can run ubiquitously on Android, IOS, and as Browser plugin with the same code. This avoids duplication of code in different platforms, lower maintenance of software and lesser bugs.

The user interface decoupled with other core sub-systems and are interfaced to communicate with the subsystems. The UI will only interfaced and will communicate with the Back end layer. UI is capable of showing user proper data, and receive actions from the user based on the user interaction and the action is converted into appropriate protocol in the decentralized application platform in order to achieve exchange on cryptocurrency. We have created User Interface to support internationalization i18n to support various languages.

4.2 Back End

The back end layer is the core part of entire platform, the process of manipulation of data, encrypting, decrypting, preparing exchange related data like Exchange pairs, Exchange price, evaluating sender/receiver address are part of Back End layer. The Back end interacts with the User interface that was explained in the previous section, this layer translates the user action into protocol level description that is explained in the following section to achieve decentralized exchange. This layer also converts the blockchain data to appropriate proof that can be easily interpreted and witnessed by the user through user interface to take correct actions to perform exchange cryptocurrency in a decentralized manner.

This layer also maintains cryptocurrency multi-wallet, creating transactions and messages, retrieving balances, creating contracts, and invoking appropriate functions in the smart contracts that are deployed on the blockchain. The Back end layer is also responsible for performing cryptographic functionality like derive public, private key from a mnemonic seed. Perform signing on the transactions, before they are broadcasted to the network, by maintaining privacy and security model of the entire platform.

This layer also communicates with the database layer, where the cached blockchain data is stored. The exchange data is sent and received between in a p2p encrypted manner, the encryption and decryption process happens in Back end layer. Once, message are received from the blockchain, the messages are decrypted by back end layer before storing it in the database.

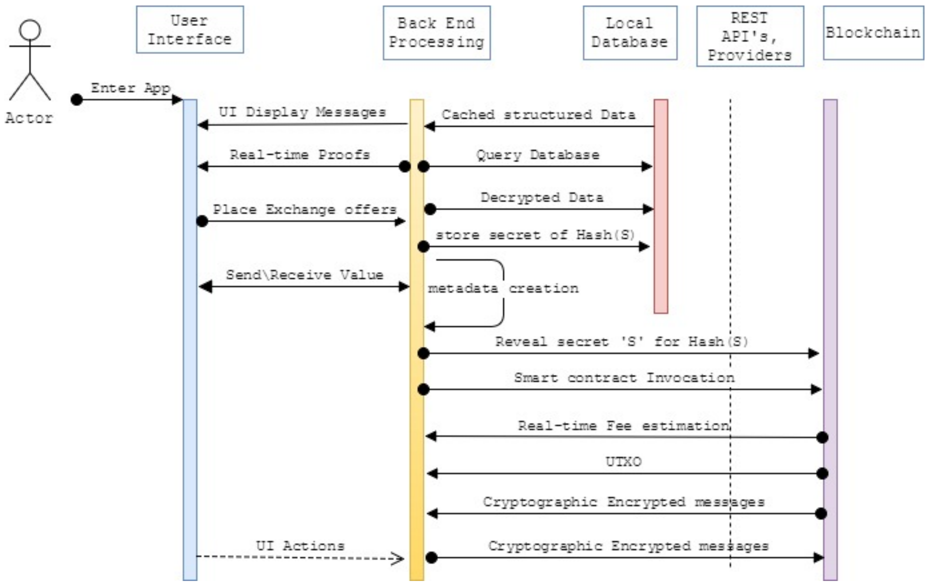
4.3 Database layer

Database layer stores cached data of the Exchange pairs, Exchange price, user specific sender/receiver data in unencrypted format. We also store proofs that reference to the appropriate blockchain in the database, proving the on-chain transaction activity to the user. All secrets are stored in the database in-order to reveal at the appropriate time. Database layer interfaces only with the Back end layer, and no other layer directly interfaces with this layer.

4.4 Blockchain

Decentralized exchange platform should be interfacing to multiple-blockchains. For this to happen, we need to interface the application to multiple blockchain without running full node on a device, as maintaining and running blockchain on a device is resource intensive. Hence, we have decided to interface the application with REST API's, also there is not a single machine where all the blockchain run. The interfacing to blockchain is done only to a public node. A public node is responsible for returning blockchain data on request from the app, the private key from the app is never exposed an public node. All transaction creation, signing are done on local device. Hence, security is preserved during the process of exchange. The public nodes are also responsible in broadcasting individual signed transactions sent by the device with respect to respective blockchain. The app will communicate with various blockchain using REST API, incase of Ethereum, the app has will invoke smart contracts with the help of provider using web3.js framework.

5 Activity diagram



Activity diagram in Decentralized exchange platform.

The above shown activity diagram shows various control flows between different layers in the decentralized exchange platform.

6 Protocol definition

Alice and Bob agrees to exchange values on Blockchain X and Y.

The problem of atomic cross-chain trading is two parties, Alice and Bob, own coins in separate cryptocurrencies, and want to exchange them without having to trust a centralized exchange.

6.1 Algorithm for Atomic Cross-Blockchain Interoperability:

1. Alice sends public offer message[a fixed public address, where order book is maintained] on independent Blockchain Z with exchange value for Blockchain

X and exchange value for Blockchain Y. Let's call this Tx.ref

Message data: Exchange values and exchange pair.

2. Bob accepts the offer by acknowledging the public offer.

   Message data:

   - Acknowledgement with Bob's pubK.X for Blockchain X with Tx.ref, proof of funds with by signing a message Tx.sig
   - Lock message to the public address in order to the lock the orderbook

3. Alice escrows value on Blockchain X after verifying T.sig proof of funds from previous transaction

   pre-requisite for escrow:

   - Randomly calculate and store secret 'S' locally and calculate hash Hash(S)
   - A default timeout period of 2 weeks for transaction expiry $\mathcal{T}_2$, after which Alice can withdraw funds to own wallet.
   - Escrow on Bob's publickey pubK.X received from previous step on Blockchain X

   Sends back message data to Bob with proof of escrow on Blockchain X

   - Alice's publickey pubK.Y on Blockchain Y
   - Hash(S) for a given random 32-byte Secret 'S'
   - Proof of escrow on the Blockchain X, that involves Bob's pubK.X
   - transaction reference Tx.ref

4. Bob witnesses proof of escrow on Blockchain X with Bob's pubK.X and Bob returns an escrow to Alice and also lock message from the public address, where only first message is accepted for the escrow not the later one's, and does the following

   Escrow's value for Blockchain Y

   - Retrieve secret hash Hash(S) which is in the previous step.
   - With a default remaining time out minus 2 days as expiry, after which Bob can withdraw funds to own wallet.
   - If $\mathcal{T}_1$ which is from step 3 is the time when offer is accepted, δ is the elapsed time since a offer is accepted, and the $\mathcal{T}_2$ return offer expiry time is calculated simply as:

$$\tau_2 = \tau_1 \text{ - } \delta \text{ - } \mathbf{k}$$

    where K is a default constant expiry time, in this case its 2 days.

    This is very important in order to avoid the counterparty risk, where in Alice tries to perform a late spend on Blockchain Y as well as quickly withdraw funds on Blockchain X which is in expiry period, there by jeopardizing Bob's turn to spend on Blockchain Y

  - Escrow on Alice's publickey pubK.Y received from the previous step on Blockchain Y.

  Sends back message to Alice with proof

  - Proof of return escrow on the Blockchain Y.
  - transaction reference Tx.ref

5. Alice witnesses the proof of escrow on Blockchain Y with Alice's publickey pubK.Y and the same Hash(S) for the given secret and Alice does the following

   Reveals secret 'S' for the hash Hash(S) and broadcasts to the network there by spending value which is received at Alice's wallet.

6. Bob witnesses the spent amount on the escrow Bob Blockchain Y by Alice, retreiveds the secret 'S' from the Blockchain Y and spends value of the escrowed amount on Blockchain X, there by receiving amount into the Bob's wallet.

   Following parts are about the *Contract Rescind* after expiry time:

7. In case Alice default's in step 4, after the given timeout/expiry Bob can withdraw funds safely in case the entire protocol is not followed.

8. In case Bob default's in step 5, after the given timeout/expiry Alice can withdraw funds safely in case the entire protocol is not followed.

   Message data that is exchanged between Alice and Bob is stored on an independent Blockchain Z, the messages metadata like trading pairs, trading price etc are encrypted, and no third party can be view the data except Alice and Bob, hence, entire exchange would preserve privacy, avoid exchange manipulations.

   Caveat:

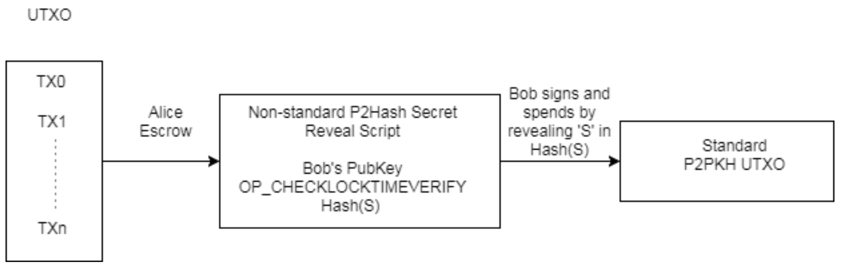
   in some cases instead of public key, address itself is used in exchange. Hash should be consistent across the blockchain, in this implementation we are using SHA256 for calculating hash with secret input.

7 Implementation details

The implementation of decentralized exchange platform requires interacting with multi-blockchain. The process of exchange involves intra-blockchain protocol mechanism, where values are exchanged without the need for trust is the goal. This is not built-in in the blockchain, but the solution to achieve the goal is to interact with blockchain with the existing framework on which the blockchains are implemented. The following sections explain how we have implemented special contracts in various blockchains:

7.1 Bitcoin blockchain:

Bitcoin's spendable balance is stored as UTXO(unspent transactions output). An UTXO can be spent as an input in a new transaction. To receive payment, the most common form is the standard Pay-To-Public-Key-Hash (P2PKH) transaction type. When Alice sends satoshi's to Bob. P2PKH lets Alice spend satoshis to a typical Bitcoin address, and then lets Bob further spend those satoshis using a simple cryptographic key pair. This is achieved through Bitcoin scripting language, the script language is a Forth-like stack-based language deliberately designed to be stateless and not Turing complete. To implement Hash Timelock Contracts, a special non-standard script where with a condition Hash(S) has to be met to spend a UTXO, such that when a secret 'S' in Hash(S) is revealed. This can be achieved in the bitcoin scripting with the combination bitcoin scripting opcode's *OP_CHECKLOCKTIMEVERIFY, OP_SHA256 and P2PKH.*


UTXO
TX0
TX1
TXn
Alice Escrow
Non-standard P2Hash Secret Reveal Script
Bob's PubKey
OP_CHECKLOCKTIMEVERIFY
Hash(S)
Bob signs and spends by revealing 'S' in Hash(S)
Standard P2PKH UTXO

showing the overview of how the process of escrow with Hash Timelock Contracts and spending of UTXO carried out between Alice and Bob.

7.2 Ethereum blockchain:

Unlike bitcoin scripting language which is non-Turing complete, Ethereum offers developers a decentralized Turing-complete virtual machine, the Ethereum Virtual Machine (EVM). We have implemented a special Hash Timelock Contracts in solidity which helps bridge exchange between Ethereum and other cryptocurrencies to achieve decentralized Atomic Cross-Blockchain Interoperability. This allows Migretor users to use the advanced smart contract features of Ethereum and it allows users of other cryptocurrencies to exchange with ease without an trusted intermediary. Ethereum virtual machine supports OPCODES in compatible with the SHA256 that we use to hash a secret 'S' to output Hash(S). The Hash Timelock Contract's deployed contract address, contract ABI interfaces programmed as part of Migretor exchange inside our application, and user will be able to invoke the contract's code directly from the mobile app or browser plug-in securely. User will be able to internally call the contract interface through app to escrow, set timeout, withdraw from the contract that is helpful in the decentralized exchange with ease. All the transactions that are interfaced with the smart contract are signed locally using private key and broadcasted. The private key never leaves your device, hence its secure.

7.3 Ardor/NXT blockchain:

NXT and Ardor blockchain takes a middle ground of what is offered in Bitcoin and Ethereum. NXT and Ardor work on Proof-of-Stake(PoS) system offers full suite of Smart Transaction templates. One of those smart transactions which is in scope and interest regarding Atomic swaps is Phasing.

Phasing is a feature that allows certain phasing-safe transactions to be created with conditional deferred execution based on the result of a vote, on a list of linked transactions or on the revelation of a secret; or simply with unconditional deferred execution. Phasing can be approved By Hashed Secret 'S' the hash of a secret phrase Hash(S), that must be revealed by the finish height to approve execution of the phased transaction. The hash can be computed from the secret phrase using SHA256 which is inconsistent with other blockchains.

All the above implementation details will have to adhere to the protocol standard described in the section XXX in order to meet decentralized Atomic Cross-Blockchain Interoperability.

8 Overcoming Sybil attack

When the exchanging is being performed, there can be fake exchange offers which needs to be discouraged, or attacker should not be allowed to perform Sybil attack on the platform to disrupt exchange services by send spam messages offering to trade. Hence, in order to avoid this, there will always be reasonable cost associated for sending messages on the blockchain. We have also kept in mind that the cost associated in messages needs to low, hence, we have chose a child chain on Ardor blockchain platform in order to store exchange messages. We will strive hard to keep the cost of exchanging the message as low as possible. We have arrived to this conclusion only after looking into different blockchains for storing data, scalability, security, speed of operation etc. As of now, the Ardor child chain seems to fit this bill.

9 Overcoming Exchange downtime

Exchange downtime can occur for various reasons, routine maintenance, exchange hack, Distributed denial of service (DDoS) to name a few. During the process of p2p decentralized cryptocurrency exchange Alice and Bob, there will be multiple messages exchanged on blockchain between the traders, every time a message is exchanged between Alice and Bob, the data like cryptocurrency pair, price quotes etc are need to be stored. This data are stored on blockchain instead of a centralized server, the advantage of storage of data in Blockchain is that, the blockchain will have greater up-time compared to any centralized server, and avoid overhead of security issues. The message data related to exchange is stored for 2 weeks and the data is pruned automatically on blockchain, there after user history will no longer be able to accessed. Hence, there needs to run special archival nodes to access the historical exchange data between the parties who have involved in exchange.

10 Privacy

During exchange Alice and Bob will not expose their public key on which exchange values are escrowed during entire exchange operation. They perform exchange using encryption as mentioned in this section

Accounts under Ardor child chains will be associated with Public keys derived from curve25519, it is used for signatures and key exchange. We always assume Alice and Bob will have their public key broadcasted on the Ardor child chain.

The main operation with those curves is the scalar multiplication during message exchange: you multiply a point P by a scalar S, and get the product PS.

$$P * S = PS$$

It is currently in-feasible to recover S even if one knows P and PS. The PS/P operation is simply prohibitively expensive without large quantum computers —or unexpected mathematical breakthrough. The division is difficult, note that the "multiplication" we are talking about is a bit like RSA's modular exponentiation. So the "division" is more like discrete logarithm.

Now the neat thing about the scalar product, is that it is associative:

$$(P * S1) * S2 = P * (S1 * S2)$$

This lets us perform the key exchange: Alice have secret key S1, bob have secret key S2. Their respective public keys are:

$$PS1 = P * S1 \text{ –Alice}$$

$$PS2 = P * S2 \text{ –Bob}$$

The shared secret between them is:

$$SS = PS1 * S2 = (P * S1) * S2 = P * (S1 * S2)$$

$$SS = PS2 * S1 = (P * S2) * S1 = P * (S2 * S1)$$

You hash the shared secret, to have a cryptographically secure random number that is shared between Alice and Bob, and secret to everyone else. You can then use that secret to encrypt message with a symmetric cipher. Ardor blockchain already implements such encryption scheme, we are simply taking advantage of this scheme in our decentralized exchange.

11 Decentralized Oracles

We need data from various blockchains such as Bitcoin, Litecoin, Ethereum. Data such as user balance, escrow data, public keys etc. These needs to be some how fed into our decentralized exchange application. Hence, initially we are planning to maintain decentralization oracles in order to feed mobile and other clients that are participating in exchange. Exchange client will communicate with oracle nodes using REST api inorder to arrive to correct conclusion while executing smart contracts and other exchange operations.

12 Decentralized OTC Public order book

A Decentralized OTC order book is maintained on blockchain. This orderbook is static and doesn't NOT perform automatic order matching, its a static orderbook as there is no centralized server to perform automatic match making engine. The orders in the order book is only executed when the user explicitly accepts the order for acknowledging it. Since, there may be multiple users acknowledging a specific order in a decentralized system, the one with the first to acknowledgment is accepted by the counterparty by comparing the timestamps on when the order is acknowledged.

13 Technology and framework

Migretor platform is built using following technology and frameworks, we are a open source platform, hence we chose to build upon source frameworks.

ANGULARJS by Google

TypeScript

node JS®

ionic

*The copyright, trademarks, logos and names belong to respective owners.

14 References

https://en.bitcoin.it/wiki/Atomic_cross-chain_trading

https://bitcoin.org/en/developer-guide
https://nxtwiki.org/wiki/Phasing