Atomic Swaps

Team members:
Ilya Selnitsky
Anastasia Smeshko
Nikita Mokrov
Emmanuel Akeweje

Abstract

Despite blockchain networks being decentralized, trading of cryptocurrencies are controlled by centralized exchanges such as Binance, Coinbase. This project explored decentralized exchange (DEX) of cryptocurrencies that allows for exchange on different blockchain networks. Hash time-locked contract, a smart contract which is locked by time and a hash function, is the main framework of the DEX method called Atomic Swaps. We implemented the algorithm for Atomic swaps using the official Python bitcoin core. Though some challenges were encountered, we explicitly explained every detail about Atomic swap, tested the implemented algorithm with the first transaction and the refund action initiated by the timelock condition and created templates for all transactions that could be decoded.

Introduction

It is not uncommon that holders of different cryptocurrencies would want to exchange their cryptocurrencies. Cross swap is the term used to describe this exchange. To illustrate cross swap, consider Alice who has Bitcoin but wants Litcoin, she then found another individual, Bob who is interested in exchanging his Litcoin for Bitcoin. Alice then asked Bob to send his Litecoin to her after which she sends her Bitcoin to him. A problem with this type of exchange is that Alice may intentionally refuse to send the Bitcoin to Bob, so she ends up with both coins. To solve this problem, a common solution is to introduce a centralized intermediate, trusted third party, to help regulate the exchange. How about having a trustless system to successfully carry out this exchange? This trustless system is Atomic Swap. An **atomic swap** is a smart contract technology that enables the exchange of one cryptocurrency for another without using centralized intermediaries, such as exchanges.

Main idea of Atomic Swaps

The idea of Atomic swap was first introduced by Tier Nolan in 2013 on the BitcoinTalk forum. The interest of the crypto community in atomic swap began to grow in September 2017 after Charlie Lee, the founder of Litecoin, announced the successful execution of atomic swap between Litecoin and bitcoin over Twitter. Cross swap can be time-consuming and complex, not all cryptocurrency exchanges support all coins. The introduction of Atomic swaps helps to mitigate these problems with the following advantages:

- Facilitation of true decentralization of Blockchain
- Mitigation of Investors' risks
- Lower transaction fees

Atomic swaps make use of Hash Timelock Contract (HTLC), a time-bound smart contract that involves the generation of cryptographic hash functions each party must verify. Atomic swaps remove risks because the parties must acknowledge receipt of funds using the cryptographic hash function within the stipulated time period; failure to do this makes the transaction void and funds reversed.

Technical Details

<u>Below is the algorithm we adapted to implement atomic swap:</u>

1. Generate addresses in each blockchain.
2. Generate random number x.
3. Calculate H(x), the hash of x.
4. A makes transaction Tx_1 and Tx_2.
    * Tx_1 has the following structure:
        * Correct input from previous blocks.
        * Condition output two cases.
            * Pay w BTC to B's public key in blockchain 1 if show preimage of H(x) (it's x).
            * If Output signed by A and B private keys.
    * Tx_2 with output Pay w BTC from Tx_1 to A, locked 48 hours in the future, signed by A.
5. A send to B Tx_2 and B signs it with its private key.
    * This transaction needs to return money back to A if B refuses to do something with Tx_1.
6. B makes transaction Tx_3 and Tx_4 in the second blockchain with the same logic as Tx_1 and Tx_2.
    * Tx_3 has the following structure
        * Correct input from previous blocks
        * Condition output two cases
            * Pay k LTC to A's public key in blockchain 2 if show preimage of H(x) (it's x)
            * If Output signed by A and B private keys
    * Tx_4 with output pay k LTC from Tx_3 to B, locked 24 hours in the future, signed by B (to return money back)
7. B sends to A Tx_4 and A signs it with its private key
8. A submits Tx_1
9. B submits Tx_3
10. A spends Tx_3 by providing x (the first case of output) now B knows x
11. B spends Tx_1 by providing x (the first case of output)

Emergency case
* A or B refuses to spend Tx_1 or Tx_3
    * We have transaction Tx_2 and Tx_4 to return money
* After revealing the x, A has 24 hours and B 48 to spend the output, to the money will be taken back by Tx_2 or Tx_4

Faced problems

With Litecoin we faced some problems. First of all, Litecoin has no working testnet sites. That is why it will be hard to test with it. So, we decided to use another fork of Bitcoin - BitCash. It gives some benefits, we can use the same script for two blockchains. We have to do just Tx_1 and Tx_2 .

These transactions have specific output and input as it was described in the algorithm. For correct exchange it is necessary to create output script for Tx_1 with two conditions: for the refund case and for case of successful payment. We have created the script for this transaction and pushed it into the blockchain. Details can be found into the attached code and presentation. The output script was decoded correctly. For using money from this output there are two possible cases. One case is payment when we need to add to input script OP_TRUE (for OP_IF checking) and secret x, revealed by the first party in case of successful exchange. The second case is returning money, and we need to add OP_FALSE to the input script and sign the transaction with two private keys (from both parties). Here we faced the problem, that input for bitcoin transactions is strictly predefined and some additional opcode usage is not allowed for standard transactions. Though the transaction with multisignature is an exception, by adding OP_FALSE the sign was not verified during pushing. Decoding of created transactions was correct, for creating scripts we used bitcoin.core tool. It seems that there should be some additional tool or additional transaction specifications that are not clearly described.

Results

First of all we received some test coins for each blockchain network: the bitcoin one and bitcoin cash.We could not find the resource for getting tesnet Litecoins, that’s why we decided on bitcoin cash. We found a way to convert bitcoin addresses to bitcoin cash, and get several coins. Due to history, they have several differences in address generation. Also, we have implemented the core transactions that correspond to payment, refund and transaction with condition to choose who should take the coins, the main library we used was an official python bitcoin core. Also, we have broadcasted some transactions and successfully decoded its outputs. However we faced the problem of correct implementation for the inputs for refund and payment transactions as the default bitcoin network has a fixed input and it’s impossible using old versions to wary an input script. The SegWit could help us, but as we mentioned before the bitcoin cash does not support this improvement. So, we’ve made the templates for all transactions that could be decoded. The result of decoding is correct one the problem was only in broadcasting. **Please look in the provided notebook for more details:** Source code

Team Contribution

All team members read extensively about atomic swaps to understand what is expected of the team, to speed up the works on the project each member was assigned to carry out the individual activities highlighted below:

Ilya Selnitsky worked assiduously on the implementation of the atomic swap algorithm on python.

Anastasia Smeshko also worked assiduously on the implementation of the atomic swap algorithm.

Nikita Mokrov made amazing presentation slides and helped with the base code algorithm.

Emmanuel Akeweje worked actively on writing the project report.

References

1. https://en.bitcoin.it/wiki/Atomic_swap
2. https://github.com/decred/atomicswap/
3. Investopedia
4. Forbes
5. Bitcoin Core