Bitcoin & Ethereum Cross-chain Atomic Swap

A Trustless Method of Exchanging Bitcoin For Ether Between Two Peers

Bachelor's Thesis submitted to the

Computer Science Engineering, Software and Multimedia developement orientation of the *Haute Ecole Arc Ingénierie (HES-SO), Switzerland*

in partial fulfillment of the requirements for the degree of

Bachelor of Applied Science in Computer Science

presented by

Luca Srdjenovic

under the supervision of

Prof. Ninoslav Marina

co-supervised by

Thomas Shababi

July 2019

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Luca Srdjenovic
Neuchâtel, 26 July 2019

Abstract

Atomic swaps are pratical for exchanging different cryptocurrencies in avoiding any trusted third-parties. This project show a swap between Bitcoin and Ethereum blockchain using payment channels tools like hashlock or timelock. When the protocol is followed by the both participants, it guarantees the swap without any risk. In the opposite, there is no scenario where someone can control both coins.

Keywords: Bitcoin, Ethereum, Atomic Swap

Acknowledgements

Thank you

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 2

Bitcoin, a peer-to-peer network

Chapter 3

Ethereum, A Decentralised Computing Platform

3.1 Test

Chapter 4

Atomic Swap, A Method of Exchanging Different Cryptocurrencies

Definition : `Atomic Swap is the process of peer-to-peer exchange of two cryptocurrencies between two parties, without using any third-party service like crypto exchange.`

In few explication, an atomic cross-chain swap is a smart contract distributed where two parties or more exchange two cryptocurrencies across different blockchains. It is called cross-chain because you are no longer dependant on the blockchain. An atomic swap protocol guarantees if both parties follow the protocol, then all swaps take place. But if one of the two parties deviates from the protocol, then no conforming party and the no coalition produce automatically the cancel of the swap. At any moment, no one can control both coins, hence no coalition has an incentive to deviate from the protocol.

4.1 Atomicity

`Atom` comes from Greek and means 'a' -not/un, 'tom' -cut, in other word, no divisible or cuttable. It means that an atomic transactions cannot be splittable into parts. We use the familiar expression `all or nothing` in atomic where it is the same applied concept in bitcoin. For example, Alice pays Bob in one transaction, they all know that either Bob will be paid or either bob won't. There is only two ways, the transaction is confirmed or not but there is no way for having an half-confirmation. That's the reason why the atomicity is fundamental in atomic swap, to protect both parties, there must be no scenario in which one part can control both coins at the same time.

An other example, no atomic transaction for illustrating is when Alice wants buy something in a web store. First, she needs to transfer the money to the site and then waits for the store send her the object back. Here there always is a chance that Alice doesn't get her purchase.

4.2 Difference with Payment Channels

In Bitcoin, Payment Channel is class of techniques designed to allow users to make multiple Bitcoin transactions without committing all of the transactions to the Bitcoin block chain. In a typical payment channel, only two transactions are added to the block chain but an unlimited or nearly unlimited number of payments can be made between the participants.[1] It is faster, cheaper transactions between parties because each transaction doesn't need to be written to the blockchain. Therefore there is only the net result of multiple transactions.

Atomic swap is not a payment channel but uses tools of it like Hashed Timelock Contracts (HTLC), a technique that can allow payments to be securely routed across multiple payment channels that we describe below (see chapter 4.3). It is a concept from the Bitcoin community that is used in the Lightning Network.

4.3 Security by Hashed Timelock Contracts

Atomic swaps uses HTLC (Hashed Timelocke Contracts), which are part of the scripting language used by most major cryptocurrencies in existence right now. Both parties involved in a cross-chain transaction submit their individual transactions to the appropriate blockchain.

HTLC is a kind of smart contracts that allows to eliminate counterparty risk using tools like hashlock and timelock. It enables time-bound transactions between the two parties. A time-bound means when a recipient at the other end of the transaction is required to acknowledge the transaction, the person needs to provide a cryptographic proof. The person also needs to provide that cryptographic proof within a time-frame. In case of failing so will automatically make the transaction null and void. In practical terms, this means that recipients of a transaction have to acknowledge payment by generating cryptographic proof within a certain timestamp. Otherwise, the transaction isn't valid. The cryptographic proof of payment that the receiver generates can then be used to trigger other actions in other payments, making HTLC a powerful technique for producing conditional payments in Bitcoin.

There are many benefits for HTLC :

1. It prevents the person who is making the payment from having to wait indefinitely to find out whether or not his or her payment goes through.
2. The person who makes the payment will not have to waste his or her money if the payment is not accepted. It will simply be returned.
3. The recipient actually helps to validate the payment on the blockchain because cryptographic proof of payment is required for the recipient to accept the payment.
4. The hashes that are created for the HTLC can be easily added to blockchains.
5. The structure of the method allows the people sending and receiving the payments do not have to trust each other or even know each other to make sure that the contract will be executed properly. In other words, each party is protected from counterparty risk.

[1] Micropayment channel: Bitcoin.org Developer Guide

To work, A Hashed Timelock Contract implements several elements from existing cryptocurrency transactions. The concept of signatures, HTLC uses multiple signatures that consists of using a private key and public key to verify and validate transactions. The main elements that make HTLC a powerful method are the concept of `hashlock` and `timelock`.

Hashlock

A hashlock is a type of encumbrance that restricts the spending of an output until a specified piece of data is publicly revealed. Hashlocks have the useful property that once any hashlock is opened publicly, any other hashlock secured using the same key can also be opened. The hashlock is a scrambled version of a cryptographic key generated by the originator of a transaction. By encumbering transaction outputs with a hashlock and timelock, the channel counterparty will be unable to outright steal funds and coins can be exchanged without outright counterparty theft.

CheckLockTimeVerify (CLTV)

In addition to hashlock, it also uses timelock. Hashed Timelock Contracts use two different types of timelocks during an Atomic Swap. The second important element of HTLC is a timelock. CheckLockTimeVerify (CLTV) uses a time base to lock and release bitcoins. This means that time constraints are hard coded and coins are released only at a specific time and date or a specific height of block size.

```
1 IF
2     <provider pubkey> CHECKSIGVERIFY
3 ELSE
4     <expiry time> CHECKLOCKTIMEVERIFY DROP
5 ENDIF
6 <client pubkey> CHECKSIG
```

Listing 4.1. Example of locking script with CheckLockTimeVerify.

CheckSequenceVerify (CSV)

The second one is CheckSequenceVerify (CSV). It is not dependent on time. Instead, it uses the number of blocks generated as a measure to keep track of when to finalize a transaction.

```
1 IF
2     <provider pubkey> CHECKSIGVERIFY
3 ELSE
4     <expiry time> CHECKSEQUENCEVERIFY DROP
5 ENDIF
6 <client pubkey> CHECKSIG
```

Listing 4.2. Example of locking script with CheckSequenceVerify.

Chapter 5

Protocol

We describe a protocol for an on-chain atomic swap between Bitcoin and Ethereum, but the protocol can be generalized for Ethereum and any other cryptocurrencies that fulfill the same requirements as Bitcoin (e.g. LiteCoin), see **??**. This protocol is heavily based on the `BIP-199` (Bitcoin Improvement Proposal (BIP)) [Bowe and Hopwood, 2017] for the Bitcoin part. For Ethereum the concept is roughly the same but with less prerequisites than Bitcoin. For sending funds, each participant must generate a specific address to lock fund on each chain (cross-chain) where each other party can take control of the funds from the other chain (swap) only.

5.1 Limitations

The most important process of the protocol is the `liveness`. Liveness means that participant must be online for respecting the protocol (at least one participant is still online). In the worst scenario where someone doesn't follow the protocol, it can happen the coalition end up and loose the funds. This happen only if a party is not remained online during the swap or it has not claimed the funds in time.

In an other side, there is an other factor to take on board which is the `Fees`. Each blockchain have different fees because there are built with different internal parameters and transaction complexity. It is also due to a factor, the blockspace that depend of the demand. In this project, we use the Bitcoin Blockchain like a tool, more precisely, we use some advanced features that increase the cost of the transaction for bitcoin side. In general, the transaction is more expensive on Bitcoin than Ethereum, because Ethereum the cost transactions doesn't depend by the user.

The difficult problem with cross chain swaps is the off chain coordination required to have the two parties meet and agree on conditions. This consist to an accord between the two peers by the speed of the protocol (i.e. to considerate that a confirmation is confirmed) but the speed is influenced with the slowness and a number of confirmation required for validating a confirmation in each blockchain side. The protocol is slow but it can be extended by way of setups. The only things we can change from the setups is the ranges of fees that can consume but in any case we cannot deviate the worst scenario that consists to have an amount of fee in each chain.

5.2 Scenario

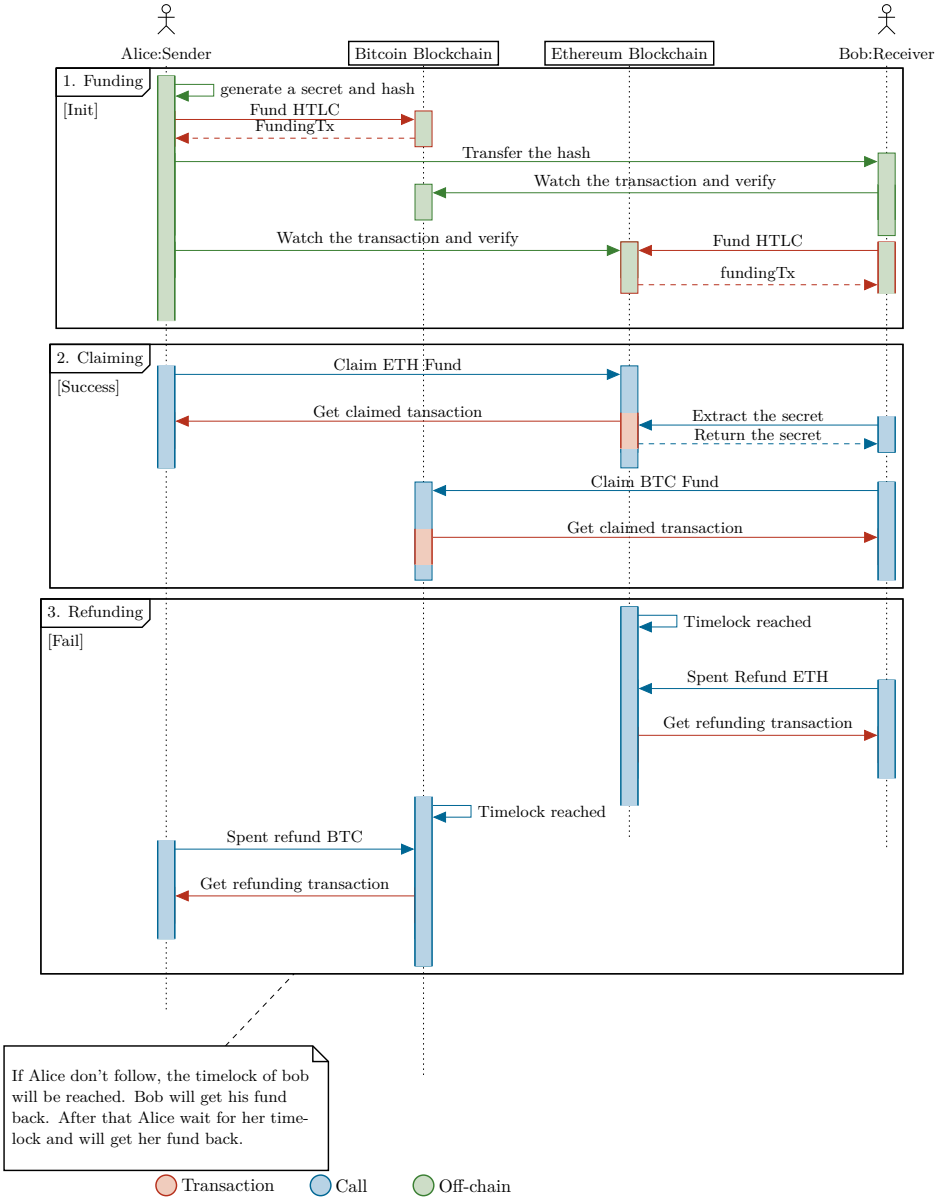

Figure 5.1. Sequence of atomic swap protocol.

Let's see the process in figure 5.1 :

- Alice owns Bitcoin (BTC) and wants to exchange Ether (ETH).
- Alice is the instigator of the swap, she starts by creating a contract address HTLC in bitcoin chain.
- Alice generate a key and hash it.

- Alice deposits the BTC and the value in the contract address. It's called the funding transaction.
- Upon doing this, Alice sends her hash to Bob.
- Bob generates too a contract address using the hash that has been given to him by Alice in ethereum chain.
- Only Alice can unlock the ETH in this address because she has the value which generates that particular hash. It's the claiming transaction.
- Alice can get her ETH by signing a transaction for Bob's contract address and Bob can retrieve the BTC by signing a transaction for Alice contract address.
- When Alice signs Bob's contract address with the value, she unlocks the address and reveals the value to Bob as well.
- Bob, now knowing the value, signs off the transaction for Alice's address and retrieves his BTC.

To summarize the process , the scenario describes the participants and their incentives. `Alice` owns Bitcoin (BTC) and Bob owns ether (ETH), they want to swap funds. Alice and Bob have already negotiated the price in advance and are agreed (i.e. amount of bitcoin for amount of ether to swap). They are only two possible ways of execution path for both parties :

1. `The protocol succeed` - Alice get her ETH and Bob his BTC.
2. `The protocol failed` - both parties keep their fund (they will lost some amounts because they need to pay some fees for each transaction).

5.2.1 Successful swap

For having a successful swap, both parties must follow the protocol. They will be four transactions in total, 2 transactions for Bitcoin blockchain and 2 transactions for Ethereum blockchain :

1. Lock the funds in Bitcoin and make it ready for the swap.
2. Lock the funds in Ethereum and make it ready for the swap.
3. Unlock the funds in Ethereum.
4. Unlock the funds in Bitcoin.

When participants unlock the funds, they take control of the output of the contract in the other chain. here is the most simple and optimal way to perform the protocol. Here, no timelock is required but both participant must care about the minimum number of transaction and for the minimal transaction, the funding transaction (locking fund). Confirmations vary between each chain, so it needs to be considered if both parties are expecting the funding transaction to be considered final and are sure to keep going the protocol.

5.2.2 Swap aborted

The swap is aborted only if one party wants not to continue the process. To get the refund of the locked funds, Alice or Bob must wait for the timelock is reached. When Alice starts, there are no ETH but only BTC locked into a contract. If bob doesn't follow, so

Alice wait her timelock and after that she can spent refund. The length of time on each lock is important to ensure that the game can only be played fairly. Alice's time lock should be longer and Bob's lock should be much shorter. This is because Alice knows the hash lock secret and therefore has a major advantage. It is very important because if Alice's timelock had the shorter refund time, Alice could wait until that time expires, refund herself the Bitcoin and after that then enter the secret preimage into Ethereum to claim the ETH that Bob sent. Alice would have both coins and Bob would loose his ETH.

5.2.3 Worst scenario

There is possibility that the protocol can be broken again if a party doesn't follow the rules from the Bitcoin part. If the swap process succeed with Alice claiming ETH funds and Bob doesn't claim his BTC fund before the Alice's timelock, then Alice can spent her refund as soon as her timelock is reached. It will conclude that Bob would lost his funds and Alice would get both coins. In Ethereum this can't happend because when the timelock is reached, claim fund are automatically blocked and Alice cannot claim the fund, only Bob spent the refund to avoid that situation. To resolve this problem, we must implement a protocol that force Bob to be not offline or compensate Bob if Alice doesn't follow correctly the protocol.

5.3 Prerequisite

In the chapter 5.2, we describe the conditional process that must be followed to guarantee a swap with atomicity. Bitcoin has a small stack-based script language that allows for conditional execution and timelocks. Whereas Ethereum use full smart contract that allows hashing and timelocks too. The challenge is then to implement the BIP-199 in Ethereum.

5.3.1 Bitcoin

5.3.2 Ethereum

5.3.3 Elliptic Curve

$$
\begin{aligned}
&p : \text{a prime number; } p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1 \\
&a : \text{an element of } \mathbb{F}_p; \ a = 0 \\
&b : \text{an element of } \mathbb{F}_p; \ b = 7 \\
&E' : \text{an elliptic curve equation; } y^2 = x^3 + bx + a \\
&G' : \text{a base point; } G' = \\
&\quad (\texttt{0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798}, \\
&\quad\ \texttt{0x483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8})
\end{aligned}
\tag{5.1}
$$

5.4 Hashed Timelock Contract

The scenario describe the participants and their incentives. `Alice`, who owns monero (XMR), and Bob, who owns bitcoin (BTC), want to swap funds. We assume that they

Alice (BTC→ETH)		Bob (ETH→BTC)
	Initialization phase	
$K_a \leftarrow k_a G$		$K_b \leftarrow k_b G$
$h_a \leftarrow \mathscr{H}_{160}(k_a)$		$h_b \leftarrow \mathscr{H}_{160}(k_b)$
$s \xleftarrow{R} \mathbb{Z}_{2^{256}}$		
$h_s \leftarrow \mathscr{H}_{256}(s)$		
	$\xleftarrow{\langle \text{send } h_b \rangle}$	
	Lock phase	
create BTX_{lock} $\langle h_a, h_b, h_s, t_0 \rangle$		
sign BTX_{lock}		
broadcast $\text{BTX}_{lock}^{signed}$		
	$\xrightarrow{\langle \text{send } h_s, \text{ watch } \text{BTX}_{lock} \rangle}$	
		verify $\text{BTX}_{lock}^{signed}$
		create ETX_{lock} $\langle h_s, t_1 \rangle$
		sign ETX_{lock}
		broadcast $\text{ETX}_{lock}^{signed}$
	$\xleftarrow{(\text{watch } \text{ETX}_{lock})}$	
verify ETX_{lock}		
	Swap phase	
unlock ETX_{lock} $\langle s \rangle$		
broadcast ETX_{unlock}		
	$\xrightarrow{\text{watch } \text{ETX}_{unlock}, \text{ extract } s}$	
		spent BTX_{lock} $\langle s \rangle$
		broadcast BTX_{unlock}

Table 5.1. Full protocol of cross-chain atomic swap between Bitcoin and Ethereum with Alice and Bob with initialization, lock, and swap phases.

already have negotiated the price in advance (i.e. amount of bitcoin for amount of monero to swap.) This negotiation can also be integrated into the protocol, for example by swap services who provide a price to their customers. Both participants wish to only have two possible execution paths (which are mutually exclusive to each other) when executing the protocol: (1) the protocol succeeds and Alice gets bitcoin, Bob gets monero, or (2) the protocol fails and both keep their original funds minus the minimum transaction fees possible. # Implementation in Bitcoin

saksaksakas

5.5 test

Chapter 6

Implementation in Ethereum

Chapter 7

Observation

Test Web3 for testing.
Test [Buterin, 2013]. # Conclusion
Conclusion

List Of Abbreviations

BIP Bitcoin Improvement Proposal. 11

HTLC Hashed Timelock Contracts. 8

Glossary

Web3 Web3 often refers to `web3js`, the Javascript implementation of the Ethereum JSON-RPC. It may also refer to other implementation in different languages. Overall it is the technology aiming to build the next and more decentralised version of the web 2.0 we know today. 19

Bibliography

Sean Bowe and Daira Hopwood. Hashed time-locked contract transactions, Mars 2017. URL `https://github.com/bitcoin/bips/blob/master/bip-0199.mediawiki`.

Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform, 2013. URL `https://github.com/ethereum/wiki/wiki/White-Paper`.