Bitcoin–Monero Cross-chain Atomic Swap

March 18, 2019 – DRAFT

h4sh3d[1]

TrueLevel
h4sh3d@truelevel.io

Abstract. Cross-chain atomic swaps have been discussed for a very long time and are very useful tools. This protocol describes how to achieve atomic swaps between Bitcoin and Monero with two transactions per chain without trusting any central authority, servers, nor the other swap participant. We propose a swap between two parties, one holding bitcoins and the other monero.

Keywords: Bitcoin, Monero, Cross-chain swaps

1 Introduction

We describe a protocol for an on-chain atomic swap between Monero and Bitcoin, but the protocol can be generalize to Monero and any other cryptocurrencies that fulfill the same requirements as Bitcoin.

This protocol is heavily based on a Monero StackExchange post discussing if it's possible to trade Monero and Bitcoin in a trustless manner [1]. The concept is roughly the same, with some changes in the Bitcoin part and is explained in more detail; they send funds to a special location on each chain (cross-chain) where each party can take control of the other chain (swap) and the other chain only (atomic).

1.1 Known limitations

To provide finality (if at least one participant is still online) we allow the worst case scenario to end up with one participant loosing his funds, but this case should only happen with negligible probability. Rationale behind this design is explained in 2.3.

Fees are different from one chain to the other because of internal blockchain parameters and transaction complexity. Because the Bitcoin blockchain is used as a decision engine transactions are, related to Bitcoin transactions, complexe and then expensive. Thus making the Bitcoin chain expensier than the Monero chain.

Speed in cross-chain atomic swap is hard to achieve, the slowest chain and the number of confirmations dictate the speed of the protocol, making front runs possible in some cases.

2 Scenario

We 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 already have negotiated the amounts of bitcoins and monero to swap in advance. This negotiation can also be integrated into the protocol for example swap services who provide a price to their customers.

Both want only to issue when executing the protocol: (1) the protocol succeed and Alice gets bitcoins, Bob gets monero, or (2) the protocol fails and both want to keep their original funds minus the minimum transactions' fees.

2.1 Successful swap

If both participants follow the protocol 4 transactions in total will be broadcast, 2 on Bitcoin blockchain and 2 on Monero blockchain. The first ones locks the funds and makes them ready for the trade on each chain. The second ones unlocks the funds for one participant only and gives knowledge to the other participant who takes control of the output on the other chain.

In that case the protocol is the fastest, only locking funds requires confirmations on each chain depending on the level of security wanted by each participant and no timelocks are waited to follow the protocol.

2.2 Swap correctly aborted

When locking the bitcoins, after a timelock Alice or Bob can start the process of refunding the locked funds. In this case the monero might not be locked yet, if no monero are locked the refund process will just refund the bitcoins, otherwise Alice will learn enough information to refund her monero too.

When the refund transaction is broadcast Bob has to spend the refund before some timelock, otherwise he might end up loosing his bitcoins without getting any monero.

2.3 Worst case scenario

If the swap is cancelled with the refund process and Bob does not spend his funds before the timelock, Alice can claim the refund. Thus one participant, Bob, end up underwater and 3 Bitcoin transactions are needed instead of 2.

Rationale This choice is made to avoid the following case: if monero are locked, Alice will be able to refund them only if Bob refund his bitcoins first, we need an incentive mechanisme to force Bob to spend his refund.

3 Prerequisites

As we described it before, conditional execution must be possible in order to achieve a swap and atomicity. Bitcoin has a small stack-based script language that allows conditional execution and timelocks. On the other hand Monero with its privacy oriented RingCT design provides only signatures to unlock UTXOs. Meaning that control of UTXOs is only related to who controls the associated private keys. The challenge is then to move control of funds only with knowledge of some private keys.

Requirements for activated features for each chain are a bit different that the StackExchange post. We describe all components needed on and off-chain.

3.1 Monero

Monero does not require any particular primitives on-chain (hashlocks, timelocks), all building blocks are off-chain primitives. Thus we need to provide proofs of the correct following of the protocol initialization, and this might be the hardest part.

Shared secret, to enable a basic two-path execution in Monero. The Monero swap private spend key is split into 2 shared secrets x_0, x_1. Participants will not use any multi-signing protocol, instead, the private spend key shares are distributed during initialization of the swap process and one participant will gain knowledge of the full key $x \equiv x_0 + x_1$ at the end of protocol execution, either for a completed swap or for an aborted swap.

Pre-image non-interactive zero-knowledge proofs of knowledge, to prove to the other participant that a valid pre-image α to a given hash $h = \mathcal{H}(\alpha)$ is known and within a specific range, e.g. $\alpha \in [1, l-1]$ where l is related to `edward25519` curve and $\mathcal{H}$ is `SHA256`. The scalar α must be serialized in a standard way.

`Edward25519` private key non-interactive zero-knowledge proofs of knowledge to prove to the other participant that a valid private key x is known for a public key X, e.g. signatures are non-interactive zero-knowledge proofs given some public keys.

3.2 Bitcoin

Bitcoin requires SegWit activated to be able to chain unbroadcast transactions. This requirements must be fulfilled for any other cryptocurrencies with UTXOs based model that might replace Bitcoin in this swap protocol.

Timelock, to enable new execution paths after some predefined amount of time, e.g. start the refund process after having locked funds on-chain without creating a race condition.

Hashlock, to reveal secrets to the other participant. Hashlocks are primitives that require to reveal some data (a pre-image) associated with a given hash to be allowed to spend the associated UTXO.

2-out-of-2 multisig, to create a common path accessible only by the two participants only if both agreed on it.

3.3 Curves parameters

Bitcoin and Monero does not use the same elliptic curves. Bitcoin use `secp256k1` curve from *Standards for Efficient Cryptography (SEC)* with `ECDSA` while Monero, based on the second version of CryptoNote, use `Curve25519`, hereinafter also `edward25519`, from Daniel J. Bernstein [2, 3]. We denote curves parameters for

`edward25519` as

$$\begin{aligned} q &: \text{a prime number; } a = 2^{255} - 19 \\ d &: \text{an element of } \mathbb{F}_q\text{; } d = -121665/121666 \\ E &: \text{an elliptic curve equation; } -x^2 + y^2 = 1 + dx^2y^2 \\ G &: \text{a base point; } G = (x, -4/5) \\ l &: \text{a base point order; } l = 2^{252} + 27742317777372353535851937790883648493 \end{aligned} \tag{1}$$

`secp256k1` as

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

4 Protocol

In this chapter we describe the protocol to follow for Alice and Bob to swap their funds atomicly and without trusting each other.

The overall protocol is as the following: Alice moves the monero into an address where each participant controls half of the private spend key. The Bitcoin scripting language is then used to reveal one or the other half of the private spend key. Depending on how reveals the shared key, the locked monero change their owner. Bitcoin transactions are designed in such a way that if a participant follows the protocol he can't terminate with a loss.

If the deal goes through, Alice spends the bitcoins by revealing her private key share, thus allowing Bob to spend the locked monero. If the deal is cancelled, Bob spends the bitcoins by revealing his private key share thus allowing Alice to spend the monero, in both cases minus transaction fees.

4.1 Time parameters

Two timelocks t_0, t_1 are defined. t_0 defines the time window during it is safe to execute the trade, after t_0 the refund process can start, making the trade unsafe to complete because of a race condition (even if hard to exploit in reallity). t_1 defines the response time during when Bob needs to react and reveals his private Monero share to get his bitcoins back and allow Alice to redeem her monero (if she locked monero).

4.2 Monero private keys

Monero private keys are pairs of `edward25519` private key. The first key is called view key and the second is called spend key. We use small letters to denote private keys and caps for public keys such that

$$X = xG$$

Where G is the generator element of the cuve (see 3.3). We denote

(i) the private key k^v as the private view key and K^v as the public view key,

(ii) k_a^v as the private view key share of Alice and k_b^v of Bob,

(iii) the private key k^s as the private spend key and K^s as the public spend key,

(iv) and k_a^s as the private spend key share of Alice and k_b^s of Bob.

Partial keys We denote partial private keys as k_a^s and k_b^s such that

$$k_a^s + k_b^s \equiv k^s \pmod{l}$$

And then

$$\begin{aligned} k_a^s G &= K_a^s \\ k_b^s G &= K_b^s \\ K_a^s + K_b^s = (k_a^s + k_b^s)G = k^s G &= K^s \end{aligned} \tag{3}$$

The same holds for k^v with k_a^v and k_b^v.

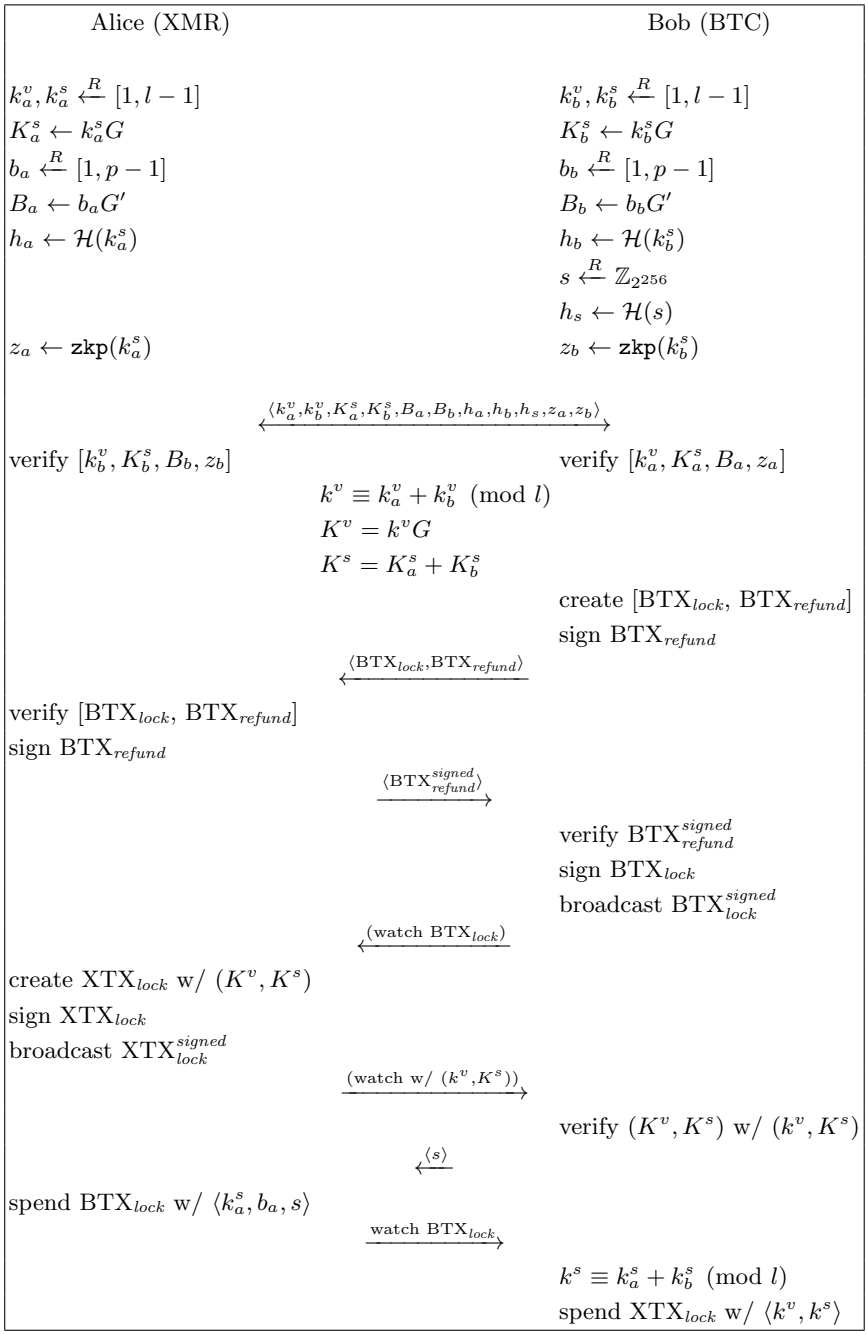

Fig. 1. Cross-chain atomic swap protocol for Bitcoin–Monero

4.3 Zero-Knowledge proofs

Zero-knowledge proofs are required at the beginning of the protocol for enabling trustlessness. The protocol uses hash as commitements of private keys, but we cannot check the hash of the other participant before it goes on-chain. Thus we need to provide a proof that the hash used in the Bitcoin script is the serialized private share and not other random data.

ZKP Alice and Bob must prove to each other with

$$\begin{aligned} k^s &= \text{valid scalar on } \texttt{edward25519} \text{ curve} \\ K^s &= k^s G \\ h &= \mathcal{H}(k^s) \end{aligned} \tag{4}$$

that given K^s and h

$$\exists k^s \mid K^s = k^s G \wedge h = \mathcal{H}(k^s) \wedge k^s \in [1, l-1] \tag{5}$$

where $\mathcal{H}(k^s)$ is the hash of the 256-bit little-endian integer representation of k^s.

4.4 Bitcoin scripts

`SWAPLOCK` is a `P2SH` used to lock funds and defines the two base execution paths: (1) swap execution and (2) refund execution.

```
OP_IF
    OP_SHA256 <h_a> OP_EQUALVERIFY
    OP_SHA256 <h_s> OP_EQUALVERIFY
    <B_a> OP_CHECKSIG
OP_ESLE
    <t_0> OP_CHECKSEQUENCEVERIFY OP_DROP
    2 <B_a> <B_b> 2 OP_CHECKMULTISIG
OP_ENDIF
```

Buy `SWAPLOCK`**,** when Alice take control of bitcoins and reveal her Monero shared key allowing Bob to take control of monero. Alice can redeem the `SWAPLOCK` with:

```
<sig_a> <s> <k_a^s> OP_TRUE <SWAPLOCK script>
```

Refund `SWAPLOCK`**,** is signed by both participants and move the funds into `REFUND P2SH`. Btx_2 use the following redeem script:

```
OP_0 <sig_a> <sig_b> OP_FALSE <SWAPLOCK script>
```

`REFUND` is an other `P2SH` used in case the swap already started on-chain but is cancelled. This refund script is used to lock the only output of a transaction that spends the `SWAPLOCK` output with the 2-out-of-2 timelocked multisig.

```
OP_IF
    OP_SHA256 <h_b> OP_EQUALVERIFY
    <B_b> OP_CHECKSIG
OP_ESLE
    <t_1> OP_CHECKSEQUENCEVERIFY OP_DROP
    <B_a> OP_CHECKSIG
OP_ENDIF
```

Spend `REFUND`, when Bob cancels the swap and reveals his Monero private share allowing Alice to regain control over her Monero. Bob can redeem the `REFUND P2SH` with:

<sig_b> <k_b^s> `OP_TRUE <REFUND script>`

Claim `REFUND`, when Alice take control of bitcoins after the second timelock without revealing her Monero shared key, ending up with Bob loosing money for not following the protocol. Alice can redeem the `REFUND P2SH` with:

<sig_a> `OP_FALSE <REFUND script>`

4.5 Transactions

$\mathbf{BTX}_{\boldsymbol{lock}}$ is a Bitcoin transaction with ≥ 1 inputs from Bob and the first output (vout: 0) to `SWAPLOCK P2SH` and optional change outputs.

$\mathbf{BTX}_{\boldsymbol{refund}}$ is a Bitcoin transaction with 1 input consuming `SWAPLOCK P2SH` (BTX_{lock}, vout: 0) with the 2-out-of-2 timelocked multisig and exaclty one output to `REFUND P2SH`.

$\mathbf{XTX}_{\boldsymbol{lock}}$ is a Monero transaction that sends fund to the address (K^v, K^s).

References

[1] PyRulez. *Can you trustlessly trade Monero for Bitcoin?* 2016. URL: `https://monero.stackexchange.com/questions/894/can-you-trustlessly-trade-monero-for-bitcoin/895#895`.

[2] Certicom Research. *SEC 2: Recommended Elliptic Curve Domain Parameters*. 2010. URL: `http://www.secg.org/sec2-v2.pdf`.

[3] Nicolas Van Saberhagen. *CryptoNote v 2.0*. 2013.