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, or the other swap participant.

Keywords: Bitcoin, Monero, Cross-chain swaps

1 Scenario

Alice, who owns Monero (XMR), and Bob, who owns Bitcoin (BTC), want to swap their funds. We assume that they already have negotiated the right amount plus some fees or what not.

They want to 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 Normal scenario

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

1.2 Worst case scenario

If the swap is cancelled, 3 Bitcoin transactions are needed instead of 2. This is to avoid a race condition that could allow Alice to gain XMR and BTC. Therefore the worst case is 5 transactions in total across both chains.

2 Prerequisites

Conditional executions must be possible in order to achieve trustless 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. That means that control of UTXOs is only related to who controls the associated one-time private keys. The challenge is then to move control of funds only with knowledge of some private keys.

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.

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

2.1 Monero

Monero does not require any particular primitives on-chain, all building blocks are off-chain primitives.

Shared secret: to enable multi-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 is split in two parts during the swap process but at the end one participant will gain knowledge of the full key $x \equiv x_0 + x_1$.

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 is known and within a range, e.g. $0 < \alpha < l$ where l is related to `edward25519` curve.

`edward25519` **private key non-interactive zero-knowledge proofs of knowledge:** to prove to the other participant that a valid private key is known, e.g. signatures are valid non-interactive zero-knowledge proof given a public key.

2.2 Bitcoin

Timelock: to enable new execution paths after some predefined amount of time, i.e. cancelling the swap even after having locked funds on-chain.

Hashlock: to reveal secrets to the other participant. Hashlocks are mini-scripts that require the sender to reveal some data (a pre-image) associated with a given hash.

2-out-of-2 multisig: to create a refund path accessible only by the two participants.

2.3 Curves parameters

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

`edward25519` as

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

`secp256k1` as

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

3 Protocol

The protocol moves XMR funds into an address (e.g. 2-out-of-2 multisig) where each participant controls half of the key. We then use the Bitcoin scripting language to reveal one or the other half of the private spend key. 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 BTC by revealing her half private key that allows Bob to spend the XMR. If the deal is cancelled, Bob spends the BTC by revealing his half private key that allows Alice to spend the XMR, both lose chain fees, in this case Bob is disadvantaged because of the 3 ”heavy” BTC transactions.

Alice (XMR)		Bob (BTC)
$a_0, x_0 \xleftarrow{R} [1, l-1]$		$a_1, x_1 \xleftarrow{R} [1, l-1]$
$X_0 \leftarrow x_0 G$		$X_1 \leftarrow x_1 G$
$b_a \xleftarrow{R} [1, p-1]$		$b_b \xleftarrow{R} [1, p-1]$
$B_a \leftarrow b_a G'$		$B_b \leftarrow b_b G'$
$h_0 \leftarrow \mathcal{H}(x_0)$		$h_1 \leftarrow \mathcal{H}(x_1)$
		$s \xleftarrow{R} \mathcal{Z}(2^{256})$
		$h_2 \leftarrow \mathcal{H}(s)$
$z_0 \leftarrow \mathtt{zkp}(x_0)$		$z_1 \leftarrow \mathtt{zkp}(x_1, s)$
	$\xleftrightarrow{\langle a_0, a_1, X_0, X_1, B_a, B_b, h_0, h_1, h_2, z_0, z_1 \rangle}$	
$\text{verify}(a_1, X_1, B_b, z_1)$		$\text{verify}(a_0, X_0, B_a, z_0)$
	$a \equiv a_0 + a_1 \pmod{l}$	
	$A = aG$	
	$X = X_0 + X_1$	
		create Btx_1, Btx_2
		sign Btx_2
	$\xleftarrow{\langle \text{Btx}_1, \text{Btx}_2 \rangle}$	
verify $\text{Btx}_1, \text{Btx}_2$		
sign Btx_2		
	$\xrightarrow{\langle \text{Btx}_2 \text{ signed} \rangle}$	
		verify Btx_2 signed
		sign Btx_1
		broadcast Btx_1
	$\xleftarrow{(\text{watch Btx}_1)}$	
create Xtx w/ (A, X)		
sign Xtx		
broadcast Xtx		
	$\xrightarrow{(\text{watch Xtx w/ } (A,X))}$	
		verify Xtx w/ (A, X)
	$\xleftarrow{\langle s \rangle}$	
spend Btx_1 w/ $\langle x_0, b_a, s \rangle$		
	$\xrightarrow{\text{watch Btx}_1}$	
		$x \equiv x_0 + x_1 \pmod{l}$
		spend Xtx w/ $\langle x, a \rangle$

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

3.1 Parameters

Two timelocks t_0, t_1 are needed. t_0 define the time window during it is sage to execute the trade, after t_0 the refund process can start. t_1 define the response time during what Bob needs to react and disclose his private Monero share.

3.2 2-out-of-2 private XMR spend key

Full XMR private key is a pair of `edward25519` private/public keys. The first pair is called view keys and the second spend keys. We use small letter to denote private keys and caps for public keys such that

$$A = aG$$

We denote

(i) the private key a as the private view key and A as the public view key,

(ii) and the private key x as the private spend key and X as the public spend key.

Partial keys We denote partial private keys as a_0 and a_1 such that

$$a_0 + a_1 \equiv a \pmod{l}$$

And then

$$\begin{aligned} a_0G &= A_0 \\ a_1G &= A_1 \\ A_0 + A_1 = (a_0 + a1)G = aG &= A \end{aligned} \tag{3}$$

The same holds for x with x_0 and x_1.

3.3 Zero-Knowledge proofs

Two zero-knowledge proofs are required at the beginning of the protocol for the trustlessness. They are quite symmetric but Bob needs to prove an extra piece of information to Alice. We denote Alice's ZKP basic ZKP and Bob's one extended ZKP.

Basic ZKP Alice must prove to Bob with

$$\begin{aligned} x_0 &= \text{valid private key on edward25519 curve} \\ X_0 &= x_0G \\ h_0 &= \mathcal{H}(x_0) \end{aligned} \tag{4}$$

that given X_0 and h_0

$$\exists x_0 \mid X_0 = x_0G \land h_0 = \mathcal{H}(x_0) \land x_0 \in [1, 2^{256}] \tag{5}$$

Extended ZKP Bob must prove to Alice with

$$\begin{aligned} x_1 &= \text{valid private key on edward25519 curve} \\ X_1 &= x_1 G \\ h_1 &= \mathcal{H}(x) \\ s &= \text{random 32 bytes data} \\ h_2 &= \mathcal{H}(s) \end{aligned} \tag{6}$$

that given X_1, h_1 and h_2

$$\exists x_1, s \mid X_1 = x_1 G \wedge h_1 = \mathcal{H}(x_1) \wedge h_2 = \mathcal{H}(s) \wedge x_1, s \in [1, 2^{256}] \tag{7}$$

3.4 Bitcoin scripts

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

```
OP_IF
    OP_SHA256 <$h_0$> OP_EQUALVERIFY
    OP_SHA256 <$h_2$> 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:

```
<$sig_a$> <$s$> <$x_0$> 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 <$h_1$> 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> <x_1> `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>`

3.5 Transactions

Btx$_\mathbf{1}$ is a Bitcoin transaction with ≥ 1 inputs from Bob and one output to `SWAPLOCK`.

Btx$_\mathbf{2}$ is a Bitcoin transaction with 1 input consuming `SWAPLOCK` with the 2-out-of-2 timelocked multisig and one output to `REFUND`.

Xtx is a Monero transaction that sends fund to the address (A, X).

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.