Enhancing Atomic Swaps with Rings Network

Ryan J. Kung
ryankung@ieee.org

April 27, 2023

Abstract

Atomic swap is a decentralized cross-chain trading method that does not require trust in third parties or centralized institutions. However, its current implementation faces many problems, and its essence is the lack of a decentralized channel, which often results in many steps of atomic swaps being performed through centralized facilities. This paper will discuss how to use the purely p2p feature of Rings Network to enhance atomic swaps, allowing them to break free from reliance on centralized facilities.

1 Introduction

Atomic swap was first introduced in 2013 [1] by Tier Nolan on the BitcoinTalk forum, and its first implementation was done by Decred and Litecoin in 2017 [2, 3]. It has since been confirmed that atomic swap can be implemented on almost all blockchain platforms, including Monero [4], Bitcoin, and Ethereum [5]. It is a trustless cross-chain method that enables decentralized trading without the need for intermediaries.

Although atomic swaps require different protocols and details to be implemented for different cryptocurrencies and blockchains (such as Monero-Bitcoin, Ethereum-Bitcoin, etc.), they can generally be broken down into three steps: negotiation, transaction and signing, and asset exchange. During the negotiation phase, participants agree upon the terms and conditions of the exchange, including the types and quantities of assets to be traded and any necessary security measures. Both participants then create and sign the transaction, and send the signature to the other party to prove their transaction intent. Once the transaction is confirmed, both participants can use the other party's hash to unlock and transfer assets to their own address.

Despite being viewed as decentralized transactions, atomic swaps rely on interactions between participants, and many steps in the process require online or offline exchanges between parties. These interactions can become centralized, which undermines the decentralized nature of the transaction. To prevent this, participants should be vigilant about maintaining their privacy and implementing protective measures to safeguard against third-party interference or centralized institutions attempting to access sensitive information during the atomic swap process.

The Rings Network is a decentralized channel that can be introduced to address these centralized issues. This browser-friendly peer-to-peer network can be compiled to run in web assembly and utilizes the Chord algorithm to construct highly scalable and high-performance real-time network channels. By leveraging the Rings Network, atomic swaps can benefit from a decentralized channel that offers enhanced privacy, security, and scalability, while also eliminating the need for centralized intermediaries.

2 Prerequisites

As atomic swaps have different implementations on various heterogeneous blockchains, it is necessary to define and differentiate them. We will define the atomic swap process between different chains using the meta-language from Grover's work [4] including InitTx(), Sign(), VrfyTx(), Vrfy(), PubTx() and RecSig().

- InitTx(): Used to initialize a transaction by setting necessary parameters.
- Sign(): Used to digitally sign a transaction using the private key of the participant.
- VrfyTx(): Used to verify the validity of a transaction by checking its input and output addresses, amounts being exchanged, and any necessary conditions.
- Vrfy(): Used to verify the validity of a digital signature by checking it against the corresponding public key.
- PubTx(): Used to publish a transaction to the blockchain, making it a permanent record.
- RecSig(): Used to receive a signature from the other party in a transaction.

2.1 Bitcoin-Monero

As shown in Figure 1, the Bitcoin to Monero transaction process can be divided into four main steps. First, Alice and Bob independently generate discrete logarithm proofs using calculations over elliptic curve groups. They then exchange and verify these proofs.

Subsequently, Bob signs the proof he received from Alice and sends it back to her. Alice verifies the signature and forwards the signed proof to Bob. Afterward, Bob proceeds to add the transaction to the blockchain. Upon receiving confirmation that the transaction has been added to the chain, Alice also generates a lock and adds it to the blockchain.

Finally, both Alice and Bob wait for on-chain notifications and complete the swap.

2.2 Bitcoin-Decred

2.3 Bitcoin-Ethereum

2.4 Bitcoin-ERC20

2.5 ERC20-ERC20

3 Implementation

4 Conclusion

abcd

References

[1] Tier Nolan. Atomic cross-chain trading, 2013.

[2] Eric Wall. Decred and litecoin complete first-ever cross-chain atomic swap. *Bitcoin Magazine*, 9 2017.

[3] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Atomic cross-chain swaps. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1–15, 2017.

[4] Sarthak Grover, Himanshu Gahlot, and Rajesh Kumar. Bitcoin-monero cross-chain atomic swap. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 244–247. IEEE, 2018.

[5] Joseph Poon and Thaddeus Dryja. Bitcoin lightning network: Scalable off-chain instant payments. *Bitcoin Lightning Network*, 2016.

Alice (XMR→BTC)		Bob (BTC→XMR)
$k_a^v, k_a^s \overset{R}{\leftarrow} [1, l-1]$		$k_b^v, k_b^s \overset{R}{\leftarrow} [1, l-1]$
$b_a, b_a^r \overset{R}{\leftarrow} [1, n-1]$		$b_b, b_b^r \overset{R}{\leftarrow} [1, n-1]$
$B_a \leftarrow b_a H$		$B_b \leftarrow b_b H$
$B_a^r \leftarrow b_a^r H$		$B_b^r \leftarrow b_b^r H$
$(z_a, K_a^s, B_a^s) \leftarrow \mathsf{DLProve}(k_a^s)$		$(z_b, K_b^s, B_b^s) \leftarrow \mathsf{DLProve}(k_b^s)$
		$s \overset{R}{\leftarrow} [0, 2^{256}]$
		$h_s \leftarrow \texttt{SHA256}(s)$
	$\xleftrightarrow{\langle k_i^v, K_i^s, B_i, B_i^s, B_i^r, z_i, h_s \rangle\ \forall i \in \{a,b\}}$	
	$k^v \equiv k_a^v + k_b^v \pmod{l}$	
	$K^v = k^v G,\ \ K^s = K_a^s + K_b^s$	
$\mathsf{DLVrfy}(K_b^s, B_b^s, z_b) \overset{?}{=} 1$		$\mathsf{DLVrfy}(K_a^s, B_a^s, z_a) \overset{?}{=} 1$
		$\texttt{BTX}_{refund})$
		$(\texttt{BTX}_{lock}, \leftarrow$
		$\mathsf{InitTx}(B_a, B_b, B_a^r, B_b^r)$
		$\sigma_r' \leftarrow \mathsf{Sign}(b_b^r, \texttt{BTX}_{refund})$
	$\xleftarrow{\langle \texttt{BTX}_{lock}, \texttt{BTX}_{refund}, \texttt{BTX}_{spend}, \sigma_r' \rangle}$	
$\mathsf{VrfyTx}(\texttt{BTX}_{lock},$		
$\texttt{BTX}_{refund}, B_a, B_b, B_a^r, B_b^r, h_s)$		
$\overset{?}{=} 1$		
$\mathsf{Vrfy}(B_b^r, \texttt{BTX}_{refund}, \sigma_r') \overset{?}{=} 1$		
$\hat{\sigma}_1' \leftarrow \mathsf{EncSign}(b_a^r, B_b^s, \texttt{BTX}_{spend})$		
$\delta' \leftarrow \mathsf{RecKey}(B_b^s, \hat{\sigma}_1')$		
$\sigma_r'' \leftarrow \mathsf{Sign}(b_a^r, \texttt{BTX}_{refund})$		
	$\xrightarrow{\langle \sigma_r'', \hat{\sigma}_1' \rangle}$	
		$\mathsf{EncVrfy}(B_a^r, B_b^s, \texttt{BTX}_{spend}, \hat{\sigma}_1') \overset{?}{=} 1$
		$\mathsf{Vrfy}(B_a^r, \texttt{BTX}_{refund}, \sigma_r'') \overset{?}{=} 1$
		$\texttt{BTX}_{buy} \leftarrow \mathsf{InitTx}(\texttt{BTX}_{lock})$
		$\hat{\sigma}_1 \leftarrow \mathsf{EncSign}(b_b, B_a^s, \texttt{BTX}_{buy})$
		$\delta \leftarrow \mathsf{RecKey}(B_a^s, \hat{\sigma}_1)$
		$\mathsf{PubTx}(\texttt{BTX}_{lock})$
	$\xleftarrow{\langle \texttt{BTX}_{buy}, \hat{\sigma}_1 \rangle}$	
$\mathsf{EncVrfy}(B_b, B_a^s, \texttt{BTX}_{buy}, \hat{\sigma}_1) \overset{?}{=} 1$		
$\mathsf{WatchTx}(\texttt{BTX}_{lock}) \overset{?}{=} 1$	$\ldots$	
$\texttt{XTX}_{lock} \leftarrow \mathsf{InitTx}(K^v, K^s)$		
$\mathsf{PubTx}(\texttt{XTX}_{lock})$	$\ldots$	
		$\mathsf{WatchTx}(K^v, K^s)$ w/ $(k^v, K^s) \overset{?}{=} 1$
	$\xleftarrow{\langle s \rangle}$	
$\sigma_1 \leftarrow \mathsf{DecSig}(k_a^s, \hat{\sigma}_1)$		
$\sigma_2 \leftarrow \mathsf{Sign}(b_a, \texttt{BTX}_{buy})$		
$\sigma := (\sigma_1, \sigma_2)$		
$\mathsf{PubTx}(\texttt{BTX}_{buy}, \sigma, s)$		
	$\ldots$	$\mathsf{WatchTx}(\texttt{BTX}_{buy}) \overset{?}{=} 1$
		$(\sigma_1, \sigma_2) \leftarrow \mathsf{RecSig}(\texttt{BTX}_{buy})$
		$k_a^s \leftarrow \mathsf{Rec}(\sigma_1, \delta)$
		$k^s \equiv k_a^s + k_b^s \pmod{l}$
		$\langle k^v, k^s \rangle$

Figure 1: Protocol execution between Alice and Bob for a successful swap [4]