Atomic swaps

Lorenzo Tucci

March 24, 2024

Parties U_0 and U_1 hold assets a on blockchain $\mathbb{A}$ and assets b on chain $\mathbb{B}$ respectively.
We define with $\mathsf{amnt_a}$ and $\mathsf{amnt_b}$ the amount of the assets the parties agreed to swap before starting the protocol.
In the protocol definition, variables and functions that are blockchain-specific (unless clear from context) are denoted with a subscript, example for a publick key on chain $\mathbb{B}$ we denote $\mathsf{pk}_{(\mathbb{B})}$.

Informally, we want the atomicity security property: parties should either both end up with the original funds in a wallet they own (refund case) or they should own the agreed assets to swap on their respective target wallets.

We define the following oracles to interact with the blockchains.

- $\mathsf{PubTx}_{(\mathbb{A})}(\sigma_{\mathsf{tx}}, \mathsf{tx})$ publish the transaction tx with signature σ_{tx} on chain $\mathbb{A}$
- $\mathsf{InitTx}_{(\mathbb{A})}(\mathsf{pk_{tx}}, \mathsf{pk_{rx}}, \mathsf{amnt})$ create an unsigned transaction paying amnt from $\mathsf{pk_{tx}}$ to $\mathsf{pk_{rx}}$ on chain $\mathbb{A}$
- $\mathsf{WatchTx}_{(\mathbb{A})}(\mathsf{tx})$ wait for the transaction tx to be confirmed on chain $\mathbb{A}$
- $\mathsf{GetBal}_{(\mathbb{A})}(\mathsf{pk})$ get the balance of assets held by pk
- $\mathsf{GetSig}_{(\mathbb{A})}(\mathsf{pk})$ get the signature σ_{tx} of the latest transaction in pk's record on chain $\mathbb{A}$

U_1 starts counting the timeout from the moment they send the VTD commitment to U_0, and respectively U_0 starts counting down from the moment they receive it.

$U_0(pk(0)\,,sk(0))$		$U_1(pk(1)\,,sk(1))$
	$\xleftrightarrow{\Gamma_{\mathsf{KeyGen}}(\mathbb{G},G,q)}$	
	$\longleftarrow (sk_0(01), pk(01))$	
	$(sk_1(01), pk(01)) \longrightarrow$	
		$(C,\pi) \leftarrow \Pi_{\mathsf{VTD}}.\mathsf{Commit}(sk_1, T)$
	$\xleftarrow{(C,\pi)}$	
starts $\mathsf{Timeout}(T-\Delta)$		starts $\mathsf{Timeout}(T-\Delta)$
if $\Pi_{\mathsf{VTD}}.\mathsf{Verify}(pk,C,\pi) \neq 1$		
abort		
$tx_{\mathsf{frz}} \leftarrow \mathsf{InitTx}(pk(0), pk(01), \mathsf{swp(a)}, \mathbb{A})$		
$\sigma_{\mathsf{frz}} \leftarrow \Pi_{\mathsf{DS}}.\mathsf{Sign}(sk(0), tx_{\mathsf{frz}})$		
$\mathsf{PubTx}(\sigma_{\mathsf{frz}}, tx_{\mathsf{frz}}, \mathbb{A})$		
starts $\Pi_{\mathsf{VTD}}.\mathsf{ForceOp}(C)$		
		do $\mathsf{bal} \leftarrow \mathsf{GetBal}(pk(01), \mathbb{A})$
		while $\mathsf{bal} \neq \mathsf{swp(a)}$
	$\xleftarrow{pk(1)}$	
$(pk(10), sk(10)) \leftarrow \Pi_{\mathsf{DS}}.\mathsf{KeyGen}(1^\lambda)$		
$tx_{\mathsf{swp}} \leftarrow \mathsf{InitTx}(pk(1), pk(10), \mathsf{swp(b)}, \mathbb{A})$		
	Γ_{Swap}	
	$U_0 \longrightarrow (sk_0(01), tx_{\mathsf{swp}})$	
	$(sk_1(01), sk(1)) \longleftarrow U_1$	
	$\longleftrightarrow$	
	$lk := \sigma_{swp}(10) \oplus sk_0(01) \longrightarrow$	
	$\longleftarrow \sigma_{swp}(10)$	
$\mathsf{PubTx}(\sigma_{\mathsf{swp(10)}}, tx_{\mathsf{frz}}, \mathbb{A})$		
		do $\sigma_{swp}(10) \leftarrow \mathsf{GetSig}(pk(1), \mathbb{B})$
		$sk(01) \leftarrow (lk \oplus \sigma_{swp}(10)) + sk_1$
		$\sigma_m \leftarrow \Pi_{\mathsf{DS}}.\mathsf{Sign}(sk(01), 1)$
		while $\Pi_{\mathsf{DS}}.\mathsf{Verify}(m, pk, \sigma_m) \neq 1$

Figure 1: Protocol execution for a successful swap (old)

Global input $(\mathbb{G}, [1], q, T, \mathsf{amnt_a}, \mathsf{amnt_b}, \mathbb{A}, \mathbb{B})$

$(\mathsf{sk_{frz0}}, \mathsf{pk_{frz}}) \leftarrow \mathbf{wait}\ \Gamma_{\mathsf{KeyGen}_{(\mathbb{A})}}(\mathbb{G}, [1], q)$
$(C, \pi) \leftarrow \mathbf{wait}\ \mathsf{receive}(U_1)$
$\mathbf{if}\ \Pi_{\mathsf{VTD}}.\mathsf{Verify}([\mathsf{pk_{frz}}] - [\mathsf{sk_{frz0}}], C, \pi) \neq 1$
$\quad \mathbf{return}\ \perp$
$\mathsf{res} \leftarrow \mathbf{select}\ \{$
$\quad \mathbf{wait}\ \{$
$\quad\quad \Pi_{\mathsf{VTD}}.\mathsf{ForceOp}(C)$
$\quad \}$
$\quad \mathbf{wait}\ \{$
$\quad\quad (\mathsf{pk_{swp}}, \mathsf{sk_{swp}}) \leftarrow \Pi_{\mathsf{DS}}.\mathsf{KeyGen}_{(\mathbb{B})}(1^\lambda)$
$\quad\quad \mathsf{tx_{frz}} \leftarrow \mathsf{InitTx}_{(\mathbb{A})}(\mathsf{pk_{init}}, \mathsf{pk_{frz}}, \mathsf{amnt_a})$
$\quad\quad \sigma_{\mathsf{frz}} \leftarrow \Pi_{\mathsf{DS}}.\mathsf{Sign}_{(\mathbb{A})}(\mathsf{sk_{init}}, \mathsf{tx_{frz}})$
$\quad\quad \mathsf{PubTx}_{(\mathbb{A})}(\sigma_{\mathsf{frz}}, \mathsf{tx_{frz}})$
$\quad\quad \mathsf{pk_{init(\mathbb{B})}} \leftarrow \mathsf{receive}(U_1)$
$\quad\quad \mathsf{tx_{swp}} \leftarrow \mathsf{InitTx}_{(\mathbb{B})}(\mathsf{pk_{init}}, \mathsf{pk_{swp}}, \mathsf{amnt_b})$
$\quad\quad \sigma_{\mathsf{swp}(\mathbb{B})} \leftarrow \Gamma_{\mathsf{Swap}}(\mathsf{sk_{frz0}}, \mathsf{tx_{swp}})$
$\quad\quad \mathsf{PubTx}_{(\mathbb{B})}(\sigma_{\mathsf{swp}}, \mathsf{tx_{swp}})$
$\quad\quad \mathsf{send}(U_1)$
$\quad \}$
$\}$
$\mathbf{if}\ \mathsf{res} \neq 1$
$\quad \mathsf{sk_{frz}} := \mathsf{sk_{frz0}} + \mathsf{res}$
$\quad \mathsf{tx_{rfnd}} \leftarrow \mathbf{wait}\ \mathsf{InitTx}_{(\mathbb{A})}(\mathsf{pk_{frz}}, \mathsf{pk_{init}}, \mathsf{amnt_a})$
$\quad \sigma_{\mathsf{rfnd}} \leftarrow \Pi_{\mathsf{DS}}.\mathsf{Sign}_{(\mathbb{A})}(\mathsf{sk_{frz}}, \mathsf{tx_{rfnd}})$
$\quad \mathbf{wait}\ \mathsf{PubTx}_{(\mathbb{A})}(\sigma_{\mathsf{rfnd}}, \mathsf{tx_{rfnd}}, \mathbb{A})$

Global input $(\mathbb{G}, G, q, T, \mathsf{amnt_a}, \mathsf{amnt_b}, \mathbb{A}, \mathbb{B})$

$(\mathsf{sk_{frz1}}, \mathsf{pk_{frz}}) \leftarrow \mathbf{wait}\ \Gamma_{\mathsf{KeyGen}_{(\mathbb{A})}}(\mathbb{G}, [1], q)$
$(C, \pi) \leftarrow \Pi_{\mathsf{VTD}}.\mathsf{Commit}(\mathsf{sk_{frz1}}, T)$
$\mathsf{send}(U_0,\ (C, \pi))$
$\mathsf{res} \leftarrow \mathbf{select}\ \{$
$\quad \mathbf{wait}\ \{$
$\quad\quad \mathsf{timeout}(T/2)$
$\quad \}$
$\quad \mathbf{wait}\ \{$
$\quad\quad \mathbf{do}\ \mathsf{bal} \leftarrow \mathsf{GetBal}_{(\mathbb{A})}(\mathsf{pk_{frz}})$
$\quad\quad \mathbf{while}\ \mathsf{bal} \neq \mathsf{amnt_a}$
$\quad\quad \mathsf{send}(U_1,\ \mathsf{pk_{init}})$
$\quad\quad lk \leftarrow \Gamma_{\mathsf{Swap}}(\mathsf{sk_{frz1}}, \mathsf{sk_{init(\mathbb{B})}})$
$\quad\quad \mathsf{receive}(U_0)$
$\quad\quad \sigma_{\mathsf{lk}} \leftarrow \mathsf{GetSig}_{(\mathbb{B})}(\mathsf{pk_{init}})$
$\quad\quad \mathsf{sk_{frz}} \leftarrow (lk \oplus \sigma_{lk}) + \mathsf{sk_{frz1}}$
$\quad\quad \mathsf{tx_{swp}} \leftarrow \mathsf{InitTx}_{(\mathbb{A})}(\mathsf{pk_{frz}}, \mathsf{pk_{swp}}, \mathsf{amnt_a})$
$\quad\quad \sigma_{\mathsf{swp}} \leftarrow \Pi_{\mathsf{DS}}.\mathsf{Sign}_{(\mathbb{A})}(\mathsf{sk_{frz}}, \mathsf{tx_{rfnd}})$
$\quad\quad \mathsf{PubTx}_{(\mathbb{A})}(\sigma_{\mathsf{swp}}, \mathsf{tx_{swp}})$
$\quad \}$
$\}$
$\mathbf{if}\ \mathsf{res} \neq 1 \wedge lk \neq \perp$
$\quad (\mathsf{pk_{rfnd}}, \mathsf{sk_{rfnd}}) \leftarrow \Pi_{\mathsf{DS}}.\mathsf{KeyGen}_{(\mathbb{B})}(1^\lambda)$
$\quad \mathsf{tx_{rfnd}} \leftarrow \mathbf{wait}\ \mathsf{InitTx}_{(\mathbb{B})}(\mathsf{pk_{init}}, \mathsf{pk_{rfnd}}, \mathsf{amnt_b})$
$\quad \sigma_{\mathsf{rfnd}} \leftarrow \Pi_{\mathsf{DS}}.\mathsf{Sign}_{(\mathbb{B})}(\mathsf{sk_{init}}, \mathsf{tx_{rfnd}})$
$\quad \mathbf{wait}\ \mathsf{PubTx}(\mathbb{B})(\sigma_{\mathsf{rfnd}}, \mathsf{tx_{rfnd}})$

Figure 2: Full protocol execution for U_0 and U_1, respectively left and right (alternative syntax)

Proof sketch

Party U_0

Informally, we want that the atomic property holds: after the protocol run either U_0 ends up with $\mathsf{amnt_b}$ on $\mathsf{pk_{swp(\mathbb{B})}}$ in case of a successful swap or with $\mathsf{amnt_a}$ on $\mathsf{pk_{init(\mathbb{A})}}$, in case the swap was aborted or refunded. We consider an active adversary over the communication channel with U_1 that can also corrupt U_1. We assume liveness and correctness for the blockchains.

By general 2PC's privacy property, $\mathsf{sk_{frz1}}$ is only known to U_0.
By the $\Pi_{\mathsf{VTD}}.\mathsf{Verify}$ algorithm we have that $\Pi_{\mathsf{VTD}}.\mathsf{Verify}(\mathsf{pk_{frz}} - [\mathsf{sk_{frz0}}], C, \pi) = 1$ if and only if the value embedded x in the commitmemt C satisfies $[x] = \mathsf{pk_{frz}} - [\mathsf{sk_{frz0}}]$.
Assuming a group based Π_{DS} with $\mathsf{pk} := [\mathsf{sk}]$, we have that $\mathsf{sk_{frz}} := \mathsf{sk_{frz0}} + \mathsf{sk_{frz1}}$ and thus $[\mathsf{sk_{frz0}} + \mathsf{sk_{frz1}}] - [\mathsf{sk_{frz0}}] = [\mathsf{sk_{frz1}}] = [x]$. Hence U_0 proceeds to swap the assets if and only if the value committed in C is $\mathsf{sk_{frz1}}$

and $\mathsf{pk}_{\mathsf{frz}} = [\mathsf{sk}_{\mathsf{frz0}} + \mathsf{sk}_{\mathsf{frz1}}]$.
Note that by the VTD's soundness property the $\mathsf{ForceOp}$ algorithm will produce the committed dlog value x in time $\mathbf{T}$, thus U_0 will be able to retrieve $\mathsf{sk}_{\mathsf{frz1}}$ after time T and sign a refund transaction.
Now U_0 transfer the funds to $\mathsf{pk}_{\mathsf{frz}}$ and proceeds to generate a new keypair $(\mathsf{pk}_{\mathsf{swp}}, \mathsf{sk}_{\mathsf{swp}})$ secret to the outside world.
When calling the 2PC protocol $\Gamma_{\mathsf{Swap}}(\mathsf{sk}_{\mathsf{frz0}}, \mathsf{tx}_{\mathsf{swp}})$, note that the inputs are again secret by 2PC's privacy property.
If $\sigma_{\mathsf{swp}(\mathbb{B})}$ is invalid, $\mathsf{PubTx}_{(\mathbb{B})}(\sigma_{\mathsf{swp}}, \mathsf{tx}_{\mathsf{swp}})$ will fail and U_0 will wait until $\mathsf{ForceOp}$ completes to compute $\mathsf{sk}_{\mathsf{frz}}$ and sign the refund transaction with it, ending up with $\mathsf{amnt}_{\mathsf{a}}$ on $\mathsf{pk}_{\mathsf{init}(\mathbb{A})}$.
Otherwise, the swap will complete successfully, and U_0 ends up with $\mathsf{amnt}_{\mathsf{b}}$ on $\mathsf{pk}_{\mathsf{swp}(\mathbb{B})}$.
An adversary has no information about $\mathsf{sk}_{\mathsf{frz0}}$ and $(\mathsf{pk}_{\mathsf{swp}}, \mathsf{sk}_{\mathsf{swp}})$, thus it cannot sign transaction from $\mathsf{pk}_{\mathsf{frz}}$ or compute a valid signature for $\mathsf{pk}_{\mathsf{swp}}$, and is thus unable to retrieve information about $\mathsf{sk}_{\mathsf{frz0}}$ from lk.

Party U_1

After the protocol run either U_1 ends up with $\mathsf{amnt}_{\mathsf{a}}$ on $\mathsf{pk}_{\mathsf{swp}(\mathbb{A})}$ in case of a successful swap, with $\mathsf{amnt}_{\mathsf{b}}$ on $\mathsf{pk}_{\mathsf{rfnd}(\mathbb{B})}$ in case the swap was refunded or $\mathsf{amnt}_{\mathsf{b}}$ on $\mathsf{pk}_{\mathsf{init}(\mathbb{B})}$ if the swap was aborted.
We consider an active adversary over the communication channel with U_0 that can also corrupt U_0. We assume liveness and correctness for the blockchains.

Note that before calling $\Gamma_{\mathsf{Swap}}(\mathsf{sk}_{\mathsf{frz1}}, \mathsf{sk}_{\mathsf{init}(\mathbb{B})})$, U_1 is in control of their assets on $\mathsf{pk}_{\mathsf{init}(\mathbb{B})}$ if it were to abort, and by general 2PC's privacy property both inputs are private.
Also note that U_1 waits until the funds $\mathsf{amnt}_{\mathsf{a}}$ have been transferred to $\mathsf{pk}_{\mathsf{frz}}$ before proceeding with Γ_{Swap}.
An adversary can only get the signature $\sigma_{swp(\mathbb{B})}$ if and only if it provided the correct $\mathsf{sk}_{\mathsf{frz0}}$ and thus U_1 has received by correctness of 2PC $lk := \sigma_{swp(\mathbb{B})} \oplus \mathsf{sk}_{\mathsf{frz0}}$.
If U_1 is unable to retrieve $\sigma_{swp(\mathbb{B})}$ before $T/2$, it proceeds to move the funds from $\mathsf{pk}_{\mathsf{init}}$ to a newly generated $\mathsf{pk}_{\mathsf{rfnd}}$, and thus an adversary holding $\sigma_{swp(\mathbb{B})}$ will be unable to get a transaction accepted by the correctness property of the blockchain (otherwise we occur in a double spending), so we end up with $\mathsf{amnt}_{\mathsf{b}}$ on $\mathsf{pk}_{\mathsf{rfnd}(\mathbb{B})}$.

If the transaction with signature $\sigma_{swp(\mathbb{B})}$ gets posted on $\mathbb{B}$, then U_1 will be able to retrieve $\mathsf{sk}_{\mathsf{frz0}}$ by the above argument, and thus compute $\mathsf{sk}_{\mathsf{frz}}$ and sign the swap transaction with it, ending up with $\mathsf{amnt}_{\mathsf{a}}$ on $\mathsf{pk}_{\mathsf{swp}(\mathbb{A})}$.

$U_0(pk(0)\ , sk(0))$	$U_1(pk(1)\ , sk(1))$
$sk(01) := sk_0(01) + sk_1(01)$	
$\sigma_{swp}(10) \leftarrow \Pi_{\mathsf{DS}}.\mathsf{Sign}(sk(1), tx_{\mathsf{swp}})$	
$lk := \sigma_{swp}(10) \oplus sk_0(01)$	

Figure 3: Protocol definition of 2PC Γ_{Swap}