Fair Atomic Cross-Chain Swaps

Yingjie Xue

Maurice Herlihy (advisor)
Outline

Motivation

Related Work

Contributions

Approaches
I want to trade my BTC for LTC

I want to trade my LTC for BTC

An exchange

No third parties?

BTC

LTC

BTC

LTC

With trusted third parties

Without any trusted third parties

Atomic Swap
No asset theft
Atomic Swap

Escrow BTC

BTC

Escrow LTC

LTC

No asset theft

Lockup griefing problem: locking the assets temporarily (24 hours) is expensive

Oh no! My BTC are locked.

Unfairness
Sore Loser Attack

- Alice’s asset depreciates, Bob walks away, Alice is stuck
- Bob’s asset depreciates, Alice walks away, Bob is stuck
- No one wants to participate.

In real life, if a party walks away, it pays a *premium*. 
Related Work

- [Komodo Platform, 2017], [Eizinger et al., 2018], [Han et al., 2019], [Goldberg et al., 2019]
- **Limitation:** asymmetric
- [Robinson, 2019]
- **Limitation:** only work for *two-party swaps*, *fungible tokens*
Contributions

✓ Proposed fair atomic swap protocols
  • Two-party atomic swap
  • Multi-party atomic swap

✓ Analyzed liveness, safety and fairness
  • Theoretical proof
  • Model checking

✓ Generalized the protocols to extreme cases
Roadmap

- Security Goals & Threat Model
- Fair Two-Party Atomic Swap
- Generalize to Extreme Cases
- Fair Multi-Party Atomic Swap
- Model Checking and Security Proofs
Security goal: Liveness

All parties conform => swap happens, premium refunded
Security goal: Safety

Conforming party => No negative payoff

premium 1$

100$ BTC

≥ 101$

Atomic Swap
Security goal: Fairness

Mitigated Risk: locking small premiums without compensation.

Assume premiums are 1%.
Threat Model

The blockchain ignores unexpected behaviors.

Deviating behaviors can be reduced to:

- Being too fast (indistinguishable from being on time)
- **Being silent** (or too slow, indistinguishable)
Roadmap

- Security Goals & Threat Model
- Fair two-party atomic swap
- Generalize to Extreme Cases
- Fair Multi-Party Atomic Swap
- Model Checking and Security Proofs
A Standard Atomic Swap Protocol

HTLC (Hashed Time-Locked Contract): a hash value $h$, and a timelock $t$. Before $t$ elapses: if a designated party inputs the preimage of $h$, they get the assets. After $t$ elapses: the assets are refunded to the owner.
Alice

BTC Blockchain

<table>
<thead>
<tr>
<th>Escrow $x_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$, lock 48 hours</td>
</tr>
</tbody>
</table>

LTC Blockchain

<table>
<thead>
<tr>
<th>Escrow $x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$, lock 24 hours</td>
</tr>
</tbody>
</table>

Bob

Time

<table>
<thead>
<tr>
<th>AS.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>redeem $x_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AS.2</th>
</tr>
</thead>
</table>

16
Contribution: Fair Atomic Swap

Alice

Deposit $p_a + p_b$, lock 96 hours

Escrow $x_1$
y, lock 48 hours

redeem $x_2$

Bob

BTC Blockchain

LTC Blockchain

Deposit $p_b$, lock 72 hours

Escrow $x_2$
y, lock 24 hours

redeem $x_1$

Deposit $p_a + p_b$, lock 96 hours

DP.1

DP.2

AS.1

AS.2

AS.3

AS.4
Alice

BTC Blockchain

LTC Blockchain

Bob

\[ p_a + p_b, 96h \]

Escrow $x_1$

\[ y, 48h \]

Redeem $x_2$

\[ y, 24h \]

Redeem $x_1$

Alice

Bob

\[
\text{lock}(p_a + p_b) \quad \text{lock}(p_a + p_b), \text{lock}(x_1), p_b \quad -p_b
\]

\[
\text{lock}(p_a + p_b), \text{lock}(x_1), x_2, p_b
\]

\[
\text{lock}(p_b), x_1
\]

\[
\text{lock}(p_a + p_b), \text{lock}(x_1) \quad \text{lock}(p_b)
\]

\[
\text{lock}(p_b)
\]

\[
-(p_a + p_b), +p_b, \text{lock}(x_1) \quad -p_b, + (p_a + p_b), \text{lock}(x_2)
\]

\[
-(p_a + p_b)\]

\[
-(x_1)
\]
Roadmap

- Security Goals & Threat Model
- Fair Two-Party Atomic Swap
- Generalize to Extreme Cases
- Fair Multi-Party Atomic Swap
- Model Checking and Security Proofs
Generalize to Extreme Cases

Repeat until premiums are acceptable small (decrease exponentially)

- Deposit Premiums of Premiums of Premiums
  - Atomic “Swap” of Premiums of Premiums
  - Atomic “Swap” of Premiums
  - Atomic Swap of Assets
Roadmap

- Security Goals & Threat Model
- Fair Two-Party Atomic Swap
- Generalize to Extreme Cases
- Fair Multi-Party Atomic Swap
- Model Checking and Security Proofs
Fair Multi-Party Swap

- A directed graph.
- First escrow and then release hashkeys
Challenges

If Alice only receives (C,A) but (B,A) is missing, Alice is stuck since:

- If Alice releases the secret, she does not get all expected assets;
- If Alice keeps the secret, she loses premiums.

Alice knows a secret and she starts depositing premiums first.
Our Proposed Approach Overview

Same recipe: deposit premiums then swap.

Two kinds of premiums:

• Escrow premiums
  • paid to the counterparty if asset is not escrowed

• Redemption premiums
  • paid to the counterparty if a hashkey is not presented to redeem the asset
Roadmap

- Security Goals & Threat Model
- Fair Two-Party Atomic Swap
- Generalize to Extreme Cases
- Fair Multi-Party Atomic Swap
- Model Checking and Security Proofs
Proof by Model Checking

- A finite-state system
- Each arc is related to a process/blockchain.
  
1. Deposit **premiums**
2. Escrow the **assets**
3. Publish **hashkeys** to redeem

Graphs used in model checking
Proof by Model Checking

- **TLA+ Language**

- **On each blockchain, steps are ordered.**
  - **In each step:**
    - Change contract states;
    - Tracking conforming status.

```plaintext
do not read this code

fair process bitcoin = BITCOIN begin
  DP2: \clock < 1, BOB deposits his premium;
  if \clock < premium["BOB"].deadline \&\& premium["BOB"].state = INIT then \other\wise, Bob cannot do anything
    premium["BOB"].state := ESCROW
    premium["BOB"].timestamp := \clock
    premium["BOB"].timeout := 2; \clock := \clock + 2, alice should escrow x1 contract["ALICE"].deadline := 2; \clock := \clock + 2, alice escrows
    step_taken[SDP2] := TRUE;
  else
    skip;
  end if;

  \* this part determines if a party is conforming
  if \step_considered[SDP1] \&\& \clock < premium["ALICE"].deadline then \* conforming[BOB] := FALSE;
  elsif \step_taken[SDP1] \&\& \step_taken[SDP2] then
    conforming[BOB] := FALSE;
  elsif \step_taken[SDP1] \&\& \step_taken[SDP2] then
    conforming[BOB] := FALSE;
  end if;

  step_considered[SDP2] := TRUE;

  AS1; \clock <= 2, Alice publishes her swap contract:
  if \step_taken[SDP2] \&\& \clock < contract["ALICE"].deadline \&\& contract["ALICE"].state = INIT then \* if DP2 finishes,
    contract["ALICE"].state := ESCROW
    contract["ALICE"].timestamp := \clock
    contract["ALICE"].timeout := 5; \clock := \clock + 5, bob redeems
    premium["BOB"].timeout := 5;
    step_taken[SAS1] := TRUE;
  elsif premium["BOB"].state = INIT then
    premium["BOB"].state := REFUNDED;
  end if;

  if \step_considered[SDP2] \&\& \clock < premium["BOB"].deadline then
    conforming[ALICE] := FALSE;
  elsif \step_taken[SDP2] \&\& \step_taken[SAS1] then
    conforming[ALICE] := FALSE;
  elsif \step_taken[SDP2] \&\& \step_taken[SAS1] then
    conforming[ALICE] := FALSE;
  end if;

  step_considered[SAS1] := TRUE;
```
Theoretical Proof

Two sequential modules: *premium* module and *swap* module

Before *swap* module starts:

- A conforming party only deposits some premiums.

After *swap* module starts:

- Safety and fairness are guaranteed by premium contracts.
Contributions & Future Work

• **First** fair atomic swaps
  • **Symmetric**
  • Two-party swap
  • Multi-party swap

• Secure proof
  • Model checking
  • Theoretical proof

• Extreme cases generalization

• Future Work
  • Other types of cross-chain transactions, e.g. auctions
  • Generic protocols
Thanks!

Questions and suggestions are welcome!
The Cox-Ross-Rubinstein (CRR) model


A brief Analysis

• If Alice deviates in DP.1, Bob has no loss
• If Bob deviates in DP.2, Alice locks($p_a + p_b$)
• If Alice deviates in AS.1, Bob locks($p_b$)
• If Bob deviates in AS.2, Alice gets $p_b$
• If Alice deviates in AS.3, Bob gets $p_a$
• If Bob deviates in AS.4, Alice gets $p_b$