Cross-chain Transactions

Crosschain workshop ’21

Mohsen Lesani,
Narges Shadab, Farzin Houshmand, Eric Chan
University of California, Riverside
Cross-chain Transactions

- An exchange protocol for general cross-chain transaction graphs with sequenced and off-chain steps.
- Guarantees uniformity and end-to-end security properties.
- A tool that automatically generates the smart contracts
- Complexity of finding the minimum leader set
Atomic Swap

Atomic Cross-Chain Swaps. Maurice Herlihy. PODC 2018
Atomic Swap

100

150

150
Atomic Swap
Atomic Swap
Atomic Swap
Atomic Swap
Atomic Swap

![Diagram showing Alice and Bob exchanging 150 Ether](image-url)
Atomic Swap
Atomic Swap
Atomic Swap

Alice

100

Bob

150

TOP SECRET
Atomic Swap
Atomic Swap
Atomic Swap
Atomic Swap

Smart Contract

100

150
Atomic Swap
Atomic Swap

Alice (100 Bitcoin) swaps for Bob (150 Ethereum)
Atomic Swap
Atomic Swap

100

TOP SECRET

150

150
Atomic Swap

100

150
Atomic Swap

Diagram:

- Alice: 150 ETH
- Bob: 100 BTC
- Lock symbol: 100 BTC
- Top Secret: 100 BTC
- Top Secret: 150 ETH
- ETH symbol
Hashed Timelock Contracts

• Locked by:
  • secret $s \rightarrow h = H(s)$
  • timeout $t$
Hashed Timelock Contracts

- Locked by:
  - secret $s$ $\rightarrow$ $h = H(s)$
  - timeout $t$
Hashed Timelock Contracts

• Locked by:
  • secret s \rightarrow h = H(s)
  • timeout t
Hashed Timelock Contracts

- Locked by:
  - secret $s \rightarrow h = H(s)$
  - timeout $t$

\[ S, h = H(s) \]

learn $S$
Hashed Timelock Contracts

• Locked by:
  • secret $s$ $\rightarrow$ $h = H(s)$
  • timeout $t$

$S, h = H(s)$
Hashed Timelock Contracts

• Locked by:
  • secret $s$  $\rightarrow$  $h = H(s)$
  • timeout $t$

\[ h = H(s) \]
Hashed Timelock Contracts

• Locked by:
  • secret $s$ $\rightarrow$ $h = H(s)$
  • timeout $t$
Hashed Timelock Contracts

• Locked by:
  • secret $s \rightarrow h = H(s)$
  • timeout $t$

Elapsed: $t$
Hashed Timelock Contracts

- Locked by:
  - secret $s$ → $h = H(s)$
  - timeout $t$

Elapsed: $t$
Atomic Swap

100

Alice

Bitcoin

Bob

LTC

200

150

Ethereum
Atomic Swap
Atomic Swap

100

TOP SECRET

#

150

200
Atomic Swap

100 BTC to 150 ETH

200 LTC to 150 ETH
Atomic Swap

100

TOP SECRET

200

Lock

150

100

200

LOCK

150
Atomic Swap

Alice

Bob

100

200

150
Atomic Swap
Atomic Swap
Atomic Swap
Atomic Swap
Cross-chain Transactions

• Cross-Chain with sequence
  • A party may need to execute steps in sequence
Cross-chain Transactions

• Cross-Chain with sequence
  • A party may need to execute steps in sequence
Cross-chain Transactions

• Cross-Chain with sequence
  • A party may need to execute steps in sequence
Cross-chain Transactions

- Cross-Chain with sequence
  - A party may need to execute steps in sequence
Cross-chain Transactions

- Cross-Chain with sequence
  - A party may need to execute steps in sequence
Cross-chain Transactions

• Cross-Chain with sequence
  • A party may need to execute steps in sequence

Not Executable
Cross-Chain with sequence

• Transformation
Cross-Chain with sequence

- Transformation
Cross-Chain with sequence

• Transformation
Cross-Chain with sequence

- Transformation
Cross-Chain with sequence

• Transformation

Diagram:

- Alice
- Ticket
- 1
- 2
- 1

Cross-Chain with sequence

• Transformation

Executable
Cross-chain Transactions

100

Off-chain step

150
Cross-chain Transactions

100

Off-chain step

150
Cross-chain Transactions

100

Off-chain step

150

TOP SECRET

Lock

Bitcoin

Ethereum

Bob the Builder
Cross-chain Transactions
Cross-chain Transactions
Cross-chain Transactions

100 BTC

Alice

150 ETH

Bob
Cross-chain Transactions
Cross-chain Transactions
Three-phase Protocol

THREE-PHASE PROTOCOL (3PP)
$P_1$  ▶ Phase 1: Contract Creation
$P_2$  if (the party is a leader)
$P_3$    Issue all the outgoing contracts.
$P_4$    Wait for all the incoming contracts and validate them;
$P_5$    on invalid incoming contracts, stop.
$P_6$    if (the party is a representative source)
$P_7$    Wait for messages from all representative sinks.
$P_8$  else ▶ the party is a follower
$P_9$    Wait for all the incoming contracts and validate them;
$P_{10}$  on invalid contracts, stop.
$P_{11}$  Issue all the outgoing contracts.
$P_{12}$  if (the party is a representative sink)
$P_{13}$  Send messages to all representative sources.
$P_{14}$  ▶ Phase 2: Release and Propagation of Secrets
$P_{15}$  if (the party is in the feedback vertex set)
$P_{16}$    Release the secret on the incoming contracts.
$P_{17}$    Apply every secret that appears on the
$P_{18}$    outgoing contracts on the incoming contracts.
$P_{19}$  ▶ Phase 3: Relay and Propagation of Secrets
$P_{20}$  if (the party is a representative source)
$P_{21}$    Upon receipt of a secret, pass it to pseudo-sinks.
$P_{22}$  else
$P_{23}$    Apply secrets that are received from sources and
$P_{24}$    from outgoing contracts to incoming contracts.
Three-phase Protocol

THREE-PHASE PROTOCOL (3PP)

$P_1$ ▷ Phase 1: Contract Creation

$P_2$ if (the party is a leader)

$P_3$ Issue all the outgoing contracts.

$P_4$ Wait for all the incoming contracts and validate them;

$P_5$ on invalid incoming contracts, stop.

$P_6$ if (the party is a representative source)

$P_7$ Wait for messages from all representative sinks.

$P_8$ else ▷ the party is a follower

$P_9$ Wait for all the incoming contracts and validate them;

$P_{10}$ on invalid contracts, stop.

$P_{11}$ Issue all the outgoing contracts.

$P_{12}$ if (the party is a representative sink)

$P_{13}$ Send messages to all representative sources.

$P_{14}$ ▷ Phase 2: Release and Propagation of Secrets

$P_{15}$ if (the party is in the feedback vertex set)

$P_{16}$ Release the secret on the incoming contracts.

$P_{17}$ Apply every secret that appears on the outgoing contracts on the incoming contracts.

$P_{19}$ ▷ Phase 3: Relay and Propagation of Secrets

$P_{20}$ if (the party is a representative source)

$P_{21}$ Upon receipt of a secret, pass it to pseudo-sinks.

$P_{22}$ else

$P_{23}$ Apply secrets that are received from sources and

$P_{24}$ from outgoing contracts to incoming contracts.
Three-phase Protocol

THREE-PHASE PROTOCOL (3PP)

\( P_1 \)  ▶ Phase 1: Contract Creation
\( P_2 \)   if (the party is a leader)
\( P_3 \)   Issue all the outgoing contracts.
\( P_4 \)   Wait for all the incoming contracts and validate them;
\( P_5 \)   on invalid incoming contracts, stop.
\( P_6 \)   if (the party is a representative source)
\( P_7 \)   Wait for messages from all representative sinks.
\( P_8 \)   else ▶ the party is a follower
\( P_9 \)   Wait for all the incoming contracts and validate them;
\( P_{10} \) on invalid contracts, stop.
\( P_{11} \) Issue all the outgoing contracts.
\( P_{12} \) if (the party is a representative sink)
\( P_{13} \) Send messages to all representative sources.
\( P_{14} \) ▶ Phase 2: Release and Propagation of Secrets
\( P_{15} \)   if (the party is in the feedback vertex set)
\( P_{16} \)   Release the secret on the incoming contracts.
\( P_{17} \)   Apply every secret that appears on the
\( P_{18} \)   outgoing contracts on the incoming contracts.
\( P_{19} \) ▶ Phase 3: Relay and Propagation of Secrets
\( P_{20} \)   if (the party is a representative source)
\( P_{21} \)   Upon receipt of a secret, pass it to pseudo-sinks.
\( P_{22} \)   else
\( P_{23} \)   Apply secrets that are received from sources and
\( P_{24} \)   from outgoing contracts to incoming contracts.
Three-phase Protocol

THREE-PHASE PROTOCOL (3PP)

\( P_1 \) ▷ Phase 1: Contract Creation
\( P_2 \) if (the party is a leader)
\( P_3 \) Issue all the outgoing contracts.
\( P_4 \) Wait for all the incoming contracts and validate them;
\( P_5 \) on invalid incoming contracts, stop.
\( P_6 \) if (the party is a representative source)
\( P_7 \) Wait for messages from all representative sinks.
\( P_8 \) else ▷ the party is a follower
\( P_9 \) Wait for all the incoming contracts and validate them;
\( P_{10} \) on invalid contracts, stop.
\( P_{11} \) Issue all the outgoing contracts.
\( P_{12} \) if (the party is a representative sink)
\( P_{13} \) Send messages to all representative sources.
\( P_{14} \) ▷ Phase 2: Release and Propagation of Secrets
\( P_{15} \) if (the party is in the feedback vertex set)
\( P_{16} \) Release the secret on the incoming contracts.
\( P_{17} \) Apply every secret that appears on the
\( P_{18} \) outgoing contracts on the incoming contracts.
\( P_{19} \) ▷ Phase 3: Relay and Propagation of Secrets
\( P_{20} \) if (the party is a representative source)
\( P_{21} \) Upon receipt of a secret, pass it to pseudo-sinks.
\( P_{22} \) else
\( P_{23} \) Apply secrets that are received from sources and
\( P_{24} \) from outgoing contracts to incoming contracts.
Leaders
Leaders

$L_1$  $L_2$  $L_3$
Three-phase Protocol

**THREE-PHASE PROTOCOL (3PP)**

$P_1$  ▷ Phase 1: Contract Creation

$P_2$  if (the party is a *leader*)

$P_3$    Issue all the outgoing contracts.

$P_4$    Wait for all the incoming contracts and validate them;

$P_5$    on invalid incoming contracts, stop.

$P_6$    if (the party is a *representative source*)

$P_7$    Wait for messages from all *representative sinks*.

$P_8$  else  ▷ the party is a *follower*

$P_9$    Wait for all the incoming contracts and validate them;

$P_{10}$  on invalid contracts, stop.

$P_{11}$  Issue all the outgoing contracts.

$P_{12}$  if (the party is a *representative sink*)

$P_{13}$  Send messages to all *representative sources*.

$P_{14}$  ▷ Phase 2: Release and Propagation of Secrets

$P_{15}$  if (the party is in the *feedback vertex set*)

$P_{16}$    Release the secret on the incoming contracts.

$P_{17}$    Apply every secret that appears on the

$P_{18}$    outgoing contracts on the incoming contracts.

$P_{19}$  ▷ Phase 3: Relay and Propagation of Secrets

$P_{20}$  if (the party is a *representative source*)

$P_{21}$    Upon receipt of a secret, pass it to *pseudo-sinks*.

$P_{22}$  else

$P_{23}$    Apply secrets that are received from sources and

$P_{24}$    from outgoing contracts to incoming contracts.
1. Contract Creation
1. Contract Creation
1. Contract Creation
1. Contract Creation
Three-phase Protocol

THREE-PHASE PROTOCOL (3PP)

$P_1$  ▷ Phase 1: Contract Creation
$P_2$  if (the party is a leader)
$P_3$    Issue all the outgoing contracts.
$P_4$  Wait for all the incoming contracts and validate them;
$P_5$  on invalid incoming contracts, stop.
$P_6$  if (the party is a representative source)
$P_7$    Wait for messages from all representative sinks.
$P_8$  else  ▷ the party is a follower
$P_9$  Wait for all the incoming contracts and validate them;
$P_{10}$  on invalid contracts, stop.
$P_{11}$  Issue all the outgoing contracts.
$P_{12}$  if (the party is a representative sink)
$P_{13}$    Send messages to all representative sources.
$P_{14}$  ▷ Phase 2: Release and Propagation of Secrets
$P_{15}$  if (the party is in the feedback vertex set)
$P_{16}$    Release the secret on the incoming contracts.
$P_{17}$  Apply every secret that appears on the
$P_{18}$  outgoing contracts on the incoming contracts.
$P_{19}$  ▷ Phase 3: Relay and Propagation of Secrets
$P_{20}$  if (the party is a representative source)
$P_{21}$    Upon receipt of a secret, pass it to pseudo-sinks.
$P_{22}$  else
$P_{23}$    Apply secrets that are received from sources and
$P_{24}$    from outgoing contracts to incoming contracts.
1. Contract Creation
Phase 1 to 2

L₁ → L₂ → L₃
Three-phase Protocol

**THREE-PHASE PROTOCOL (3PP)**

\[ P_1 \mapsto \text{Phase 1: Contract Creation} \]

\[ P_2 \quad \text{if (the party is a leader)} \]

\[ P_3 \quad \text{Issue all the outgoing contracts.} \]

\[ P_4 \quad \text{Wait for all the incoming contracts and validate them;} \]

\[ P_5 \quad \text{on invalid incoming contracts, stop.} \]

\[ P_6 \quad \text{if (the party is a representative source)} \]

\[ P_7 \quad \text{Wait for messages from all representative sinks.} \]

\[ P_8 \quad \text{else} \mapsto \text{the party is a follower} \]

\[ P_9 \quad \text{Wait for all the incoming contracts and validate them;} \]

\[ P_{10} \quad \text{on invalid contracts, stop.} \]

\[ P_{11} \quad \text{Issue all the outgoing contracts.} \]

\[ P_{12} \quad \text{if (the party is a representative sink)} \]

\[ P_{13} \quad \text{Send messages to all representative sources.} \]

\[ P_{14} \mapsto \text{Phase 2: Release and Propagation of Secrets} \]

\[ P_{15} \quad \text{if (the party is in the feedback vertex set)} \]

\[ P_{16} \quad \text{Release the secret on the incoming contracts.} \]

\[ P_{17} \quad \text{Apply every secret that appears on the} \]

\[ P_{18} \quad \text{outgoing contracts on the incoming contracts.} \]

\[ P_{19} \mapsto \text{Phase 3: Relay and Propagation of Secrets} \]

\[ P_{20} \quad \text{if (the party is a representative source)} \]

\[ P_{21} \quad \text{Upon receipt of a secret, pass it to pseudo-sinks.} \]

\[ P_{22} \quad \text{else} \]

\[ P_{23} \quad \text{Apply secrets that are received from sources and} \]

\[ P_{24} \quad \text{from outgoing contracts to incoming contracts.} \]
2. Release and Propagation of Secrets
2. Release and Propagation of Secrets

L₁

L₂

L₃
2. Release and Propagation of Secrets
2. Release and Propagation of Secrets
2. Release and Propagation of Secrets
2. Release and Propagation of Secrets
Three-phase Protocol

THREE-PHASE PROTOCOL (3PP)

- Phase 1: Contract Creation
  - $P_1$  ➤  if (the party is a leader)
  - $P_2$  ➤  Issue all the outgoing contracts.
  - $P_3$  ➤  Wait for all the incoming contracts and validate them;
  - $P_4$  ➤  on invalid incoming contracts, stop.
  - $P_5$  ➤  if (the party is a representative source)
  - $P_6$  ➤  Wait for messages from all representative sinks.
  - $P_7$  ➤  else ➤  the party is a follower
  - $P_8$  ➤  Wait for all the incoming contracts and validate them;
  - $P_9$  ➤  on invalid contracts, stop.
  - $P_{10}$ ➤  Issue all the outgoing contracts.
  - $P_{11}$ ➤  if (the party is a representative sink)
  - $P_{12}$ ➤  Send messages to all representative sources.
  - $P_{13}$ ➤  else ➤  the party is a follower

- Phase 2: Release and Propagation of Secrets
  - $P_{14}$ ➤  if (the party is in the feedback vertex set)
  - $P_{15}$ ➤  Release the secret on the incoming contracts.
  - $P_{16}$ ➤  Apply every secret that appears on the
  - $P_{17}$ ➤  outgoing contracts on the incoming contracts.
  - $P_{18}$ ➤  else ➤  the party is a follower

- Phase 3: Relay and Propagation of Secrets
  - $P_{19}$ ➤  if (the party is a representative source)
  - $P_{20}$ ➤  Upon receipt of a secret, pass it to pseudo-sinks.
  - $P_{21}$ ➤  else
  - $P_{22}$ ➤  Apply secrets that are received from sources and
  - $P_{23}$ ➤  from outgoing contracts to incoming contracts.
  - $P_{24}$ ➤  from outgoing contracts to incoming contracts.
2. Release and Propagation of Secrets

L₁ → L₂ → L₃
3. Relay and Propagation of Secrets
Release and Propagation of Secrets
Release and Propagation of Secrets
Release and Propagation of Secrets

\[ L_3 \]

\[ L_1 \]

\[ L_2 \]
Release and Propagation of Secrets

L₁, L₂, L₃
Release and Propagation of Secrets

$L_1$ $L_2$ $L_3$
Release and Propagation of Secrets
Release and Propagation of Secrets

L₃

L₁

L₂
Release and Propagation of Secrets

$L_1\rightarrow L_2 \rightarrow L_3$
Release and Propagation of Secrets
Guarantee Theorems

If the representative sources are conforming

Uniformity

• If all parties follow the protocol, all the assets should be transferred.
• If any party deviates from the protocol, the conforming parties should not experience a loss.

End-to-end

• If the source parties pay, the sink parties are paid.
XChain tool

Transaction Graph → XChain → Smart Contracts for all parties in Solidity
new offers. This process may repeat multiple times.

Bob want to increase the deposit of their contracts to 5 ethers and 4 bitcoins respectively. Thus, the next round starts with the incoming contract to get the Cadilac and Alice learns the secret and applies it to the contract coming from David to pass her Cadilac to him. Then, David applies his secret to then David wins the auction as shown in Fig. 5b. In this case, the highest offer. Let’s assume that the highest bid of this it to Alice. Alice compares the suggested bids and selects

<table>
<thead>
<tr>
<th>#</th>
<th>#Nodes</th>
<th>#Leaders</th>
<th>Synthesis Time (ms)</th>
<th>Average Gas</th>
<th>Average Price (Ether)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>294</td>
<td>1400331</td>
<td>0.02801</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>324</td>
<td>1440434</td>
<td>0.02881</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>330</td>
<td>1722722</td>
<td>0.03445</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
<td>310</td>
<td>1520925</td>
<td>0.03042</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>2</td>
<td>318</td>
<td>1673709</td>
<td>0.03347</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>2</td>
<td>343</td>
<td>1716544</td>
<td>0.03433</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>4</td>
<td>351</td>
<td>2070887</td>
<td>0.04142</td>
</tr>
</tbody>
</table>
Brokering with HTLCs
Flower Graphs and Brokering

**Brokering:** parties can leverage assets they gain during the transaction.  
**Flower Graph:** A special type of transaction graph with brokering with the following properties:

- exactly one broker
- only contracts between the broker and other parties

**Motivation:** Flower graphs replicate a market clearing service arranging a brokering scenario between many users

**Result:** Finding a minimum leader set for a flower graph is NP-Hard
Brokering with HTLCs

Brokering is the ability for parties to leverage assets they gain during the transaction. Suppose $b$ owns no assets.

Without loss of generality, let $b$ and $X$ be leaders:

- $X$ creates the contract of $1B$ to $b$, who creates the contract of $1B$ to $Y$
- $Y$ creates the contract of $1E$ to $b$, who creates the contract of $1E$ to $Z$
- $Z$ creates the contract of $3Z$ to $b$, who creates the contract of $2Z$ to $Z$
Minimum Leader Set

- Global gas cost is a function of number of leaders

- A feedback vertex set (FVS) is necessary and sufficient for a leader set (Herlihy ’18)

- Arbitrary FVS not sufficient for transactions with brokering (e.g. b as the sole leader)
Flower Graphs

- Exactly one broker
- Contracts only between broker and other parties
- No contracts between these other parties

Motivation: replicates the behavior of a market clearing service acting as the broker
Flower Graphs are NP-Hard

Reduce decision-Knapsack $K = (X, V, W)$ to finding a minimum leader set of a flower graph $F = (\mathcal{V}, \mathcal{E})$. We want the property that:

$$F \text{ has a leader set of size 2 } \iff \exists R \subseteq X \text{ s.t. } \sum_{(v_i, w_i) \in R} v_i \geq V \text{ and } \sum_{(v_i, w_i) \in R} w_i \leq W$$
Constructing $F$

- $m = \left\lceil \frac{\sum_{(v_i, w_i) \in X} w_i}{\min_w (X) - 1} \right\rceil + 1$

- Currencies: $B, E, D, C_1 \ldots C_m$

Intuition: $s$ will give $b$ $W$ $B$ to solve the knapsack problem towards $x_1 \ldots x_n$

Other nodes are used to create all remaining contracts to parties not chosen in $R$
NP-Hardness

Lemma 1: If $K$ is satisfiable, $F$ has a leader set of size 2 (the minimum)

Lemma 2: If $F$ has a leader set of size 2, then it must be $\{b, s\}$

($\Leftarrow$) If $K$ is satisfiable with $R$, then we can choose $\{b, s\}$ to be our leader set. The broker $b$ creates contracts with all nodes in $R$, then completes the transaction.

($\Rightarrow$) Since $\{b, s\}$ must be our leader set, $b$ must first create contracts with the item nodes. Additional bitcoins cannot be obtained by $b$ until a total of $VE$ has been obtained. This guarantees that the first set of item nodes that $b$ gives bitcoins to obtain $VE$ also satisfies $K$