ECS 265 Paper Presentation

Atomic Commitment Across Blockchains

Authors: Victor Zakhary, Divyakant Agrawal, Amr El Abbadi

Presented by Aadarsh Venugopal, Bharath Kinnal, Yathesh Lekkalapudi

What is Cross-Chain Transaction?

- **Cross-Chain Transaction :** Distributed transaction which spans across multiple blockchains to transfer ownership of assets
  - Consist of sub-transactions which transfers asset on some blockchain
- **Atomic Cross-Chain Transaction :** Transaction happens or doesn’t happen

What is Cross-Chain Transaction?

Cross-Chain Transaction

Example : Exchange BTC with ETH

Original Process :

- Find an TRusted cENTralized exchange (TRENT)
- Sign up and get verified
- Deposit Bitcoin to the exchange
- Trade Bitcoin with Ether for a transaction fee at the centralized exchange
- Withdraw Ether from the exchange with a service fee (withdrawal fees)

Issues with original process

- Dependence on central exchange for all cross-chain transactions
  - No longer decentralized
- Each party required to trust central exchange
  - No longer trust-free
- Transaction requires parties to pay transaction fees
- Increases the number of transactions to achieve intended cross-chain transaction

This necessitates need for a **trust-free** and **decentralized atomic** cross-chain transaction protocol

Architectural Overview

Permissionless blockchain system consists of two parts :

- **Storage Layer**
  - Decentralized distributed ledger (**blockchain**)
  - Computing nodes (**miners**) who validate and add transactions to blockchain
- **Application Layer**
  - End-users who communicate with storage layer through message passing
  - **End-users generate transactions**, which are then **validated by miners**

Blockchain stores two important info :

- Ownership information of **assets** (stored as end-user’s public key)
- **Transactions** that transfer ownership of assets between end-users

Smart Contracts

- Program sent by end-users and executed by miners, containing
  - Smart contract code
  - Sender’s and receiver’s public key
  - Asset
- Implemented as a class object
  - Contract state
    - **Published** : Contract published by end-user initiating transaction
    - **Redeemed** : If transaction is **committed**, then end-users redeem their respective rewards of transaction
    - **Refunded** : If transaction is **aborted**, then end-users get refunded their original amount specified in transaction
  - Set of functions that alter the state

Atomic Swap [Nolan ’13, Herlihy ’18]

Cross-chain transaction enabled using

- Smart contracts
- Hashlocks
    - Assets in the smart contracts are locked by one way cryptographic hash function using a secret code.
    - The assets can be unlocked only if the secret code is provided.
- Timelocks
    - Time bounded locks which triggers a smart contract function if the time expires.

Atomic Swap [Nolan ’13, Herlihy ’18]

- Alice wants to trade Bitcoin for Ethereum with Bob

Atomic Swap [Nolan ’13, Herlihy ’18]

- Alice wants to trade Bitcoin for Ethereum with Bob

- Create a secret s
- Calculate its hash **h = H(s)**

Bob

s and h

Alice

Atomic Swap [Nolan ’13, Herlihy ’18]

- Alice wants to trade X Bitcoin for Y Ethereum with Bob

SC_1 : Move X bitcoins to Bob if Bob provides secret **s | h = H(s)**

Refund SC_1 to Alice if Bob does not execute SC_1 before **48** hours

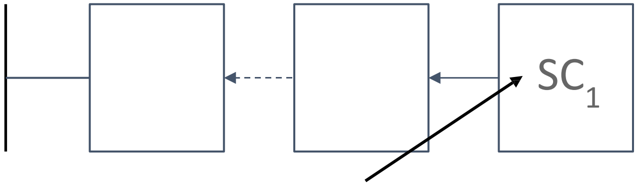
Bob

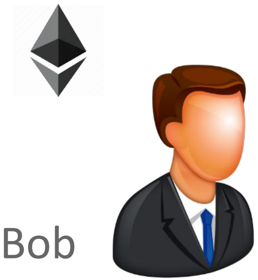
Alice

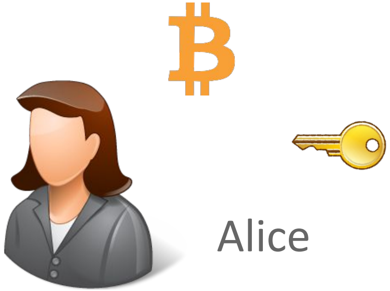
s and h

Atomic Swap [Nolan ’13, Herlihy ’18]

- Alice wants to trade X Bitcoin for Y Ethereum with Bob



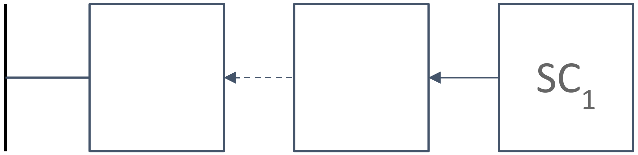




Atomic Swap [Nolan ’13, Herlihy ’18]

- Now, h is announced in Bitcoin blockchain and made public

Bitcoin blockchain



SC_2 Move Y Ethereum to Alice if Alice provides secret **s | h = H(s)**

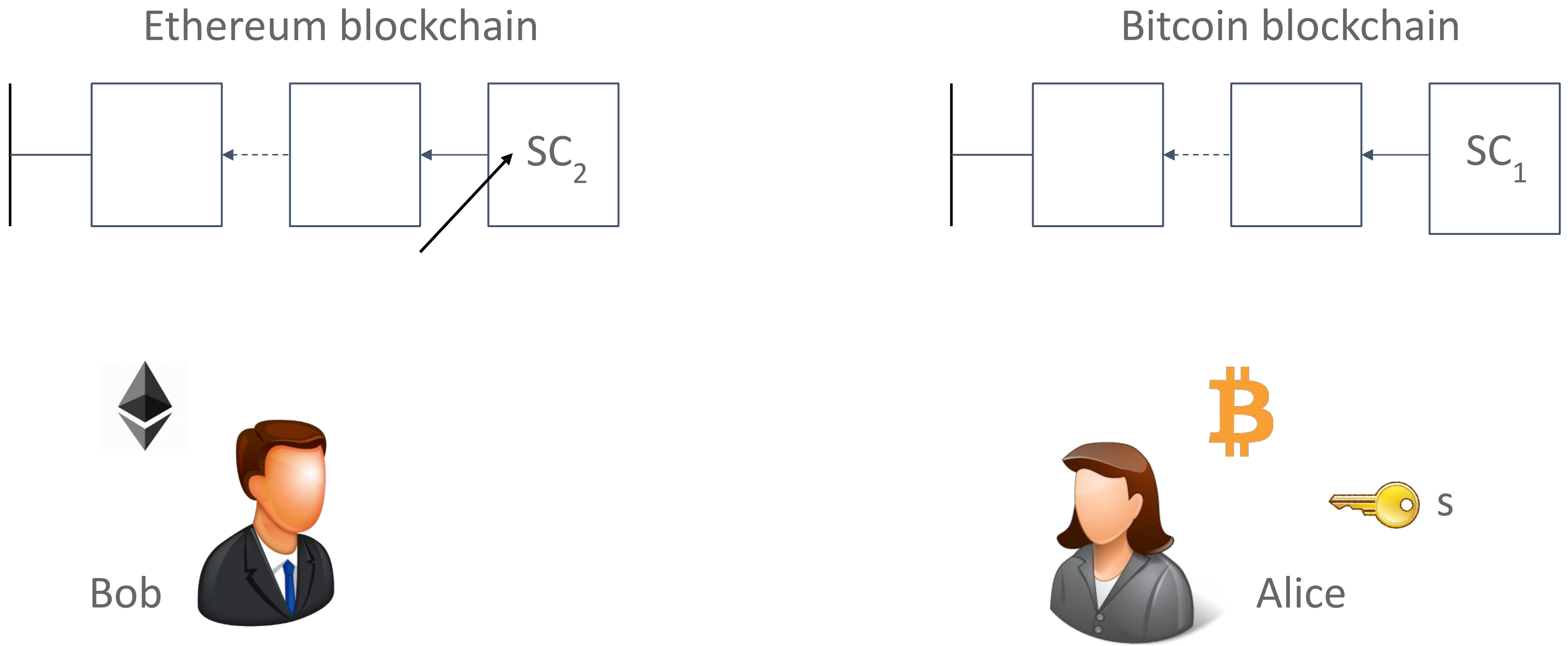
Refund SC_2 to Bob if Alice does not execute SC_2 before **24** hours

Bob

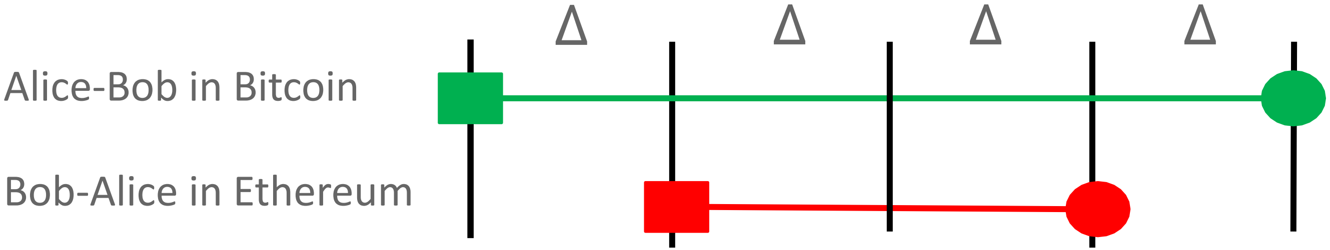
s

Alice

Atomic Swap [Nolan ’13, Herlihy ’18]

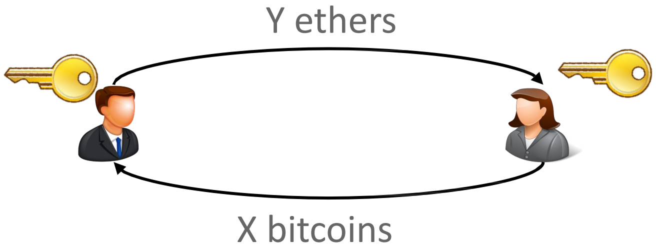


Atomic Swap Example [Nolan ’13, Herlihy ’18]



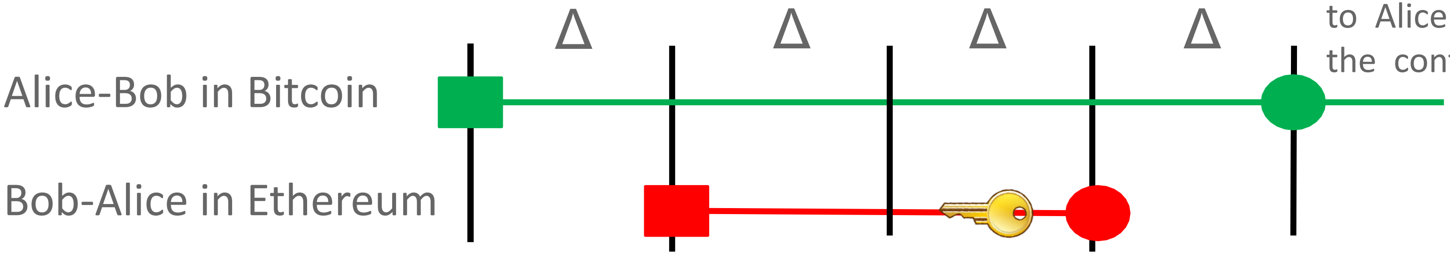
Alice reveals the secret to Bob’s contract and claims the Y ether

Supposedly, Bob takes the secret, reveals it to Alice’s contract and claims the X bitcoins



e.g., Δ = 12hr

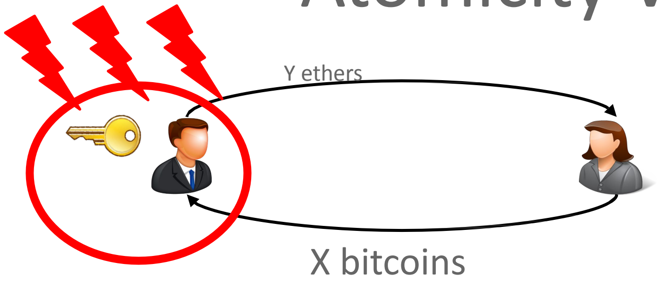
Drawback



X bitcoins are refunded to Alice any time after the contract expires

If Bob fails or suffers a network denial of service attack for a Δ, Alice's contract will expire and Bob will lose his X bitcoins

Atomicity Violation



e.g., Δ = 12hr

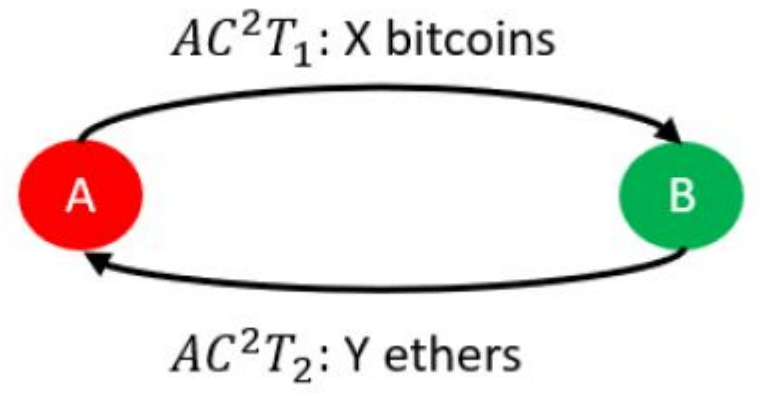
Why need another Atomic Cross-Chain Transaction protocol

- Atomicity
  - An Atomic cross- chain transaction protocol consists of bunch of sub transactions. Either all of the sub-transactions need to commit or all of the subtransactions need to abort.
  - The protocol ensures that atomicity is never violated.
- Commitment
  - Atomic cross chain protocol can decide upon a commitment of either to commit or abort sub-transactions
  - Once the commitment to a transaction is made the commitment should eventually happen.

Atomic Cross Chain Transaction Model

Atomic Cross Chain Transaction can be modeled as a directed graph D = (V,E).

- V represents the set of participants associated with transaction.
- E represents the set of all sub-transactions among participants.
    - Sub-transaction e = (u,v) means that u is the sender, and v is recipient



Atomic Cross Chain Transaction Model

AC^2T - Atomic Cross Chain Transaction.

- For every edge (u,v) a smart contract is deployed in **published state**
  - Deployed smart contract represents participant who wants to go ahead with the transaction.
- Smart contract has set of commitment schemes
  - **Redeem Commitment Scheme :** When all participants agrees, protocol commits transaction
  - **Refund Commitment Scheme :** If even one participant disagrees, protocol aborts transaction

Cross-Chain Validation of Transaction

Validation system involves :

- **Validated Blockchain :** The blockchain where state of current smart contract needs to be validated
  - **Last stable block :** Block at depth **d** from current head of validated blockchain, where probability of forking is negligible
- **Validator :** All miners for one blockchain validate cross-chain transactions in other blockchains
  - Also known as witness

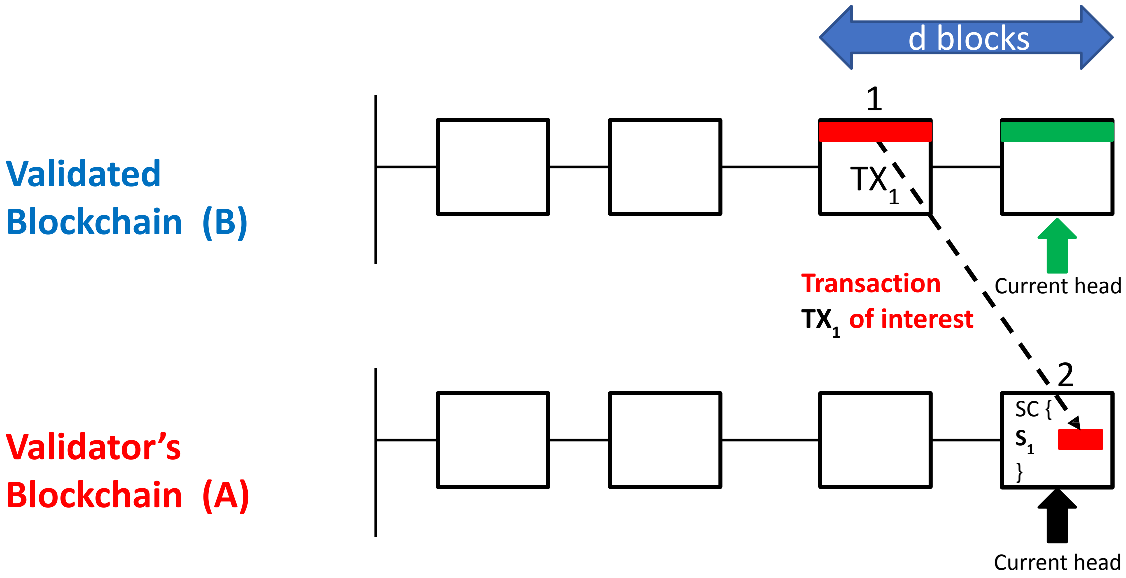
Cross-Chain Validation of Transaction

Implementation I : Simply **duplicate** all blockchains across all validators

- Highly inefficient, impractical
- Requires significant storage and network capabilities

Cross-Chain Validation of Transaction

Validator A needs to **validate** TX_1 in **validated** blockchain of miner B



Blockchain B is duplicated

Cross-Chain Validation of Transaction

Implementation 2 : Validators to run **light nodes**, which have

- Stores block headers of validated blockchains
- Verifies PoW of these block headers
- Downloads only blockchain branches associated with transaction of interest to this node

Miner stores :

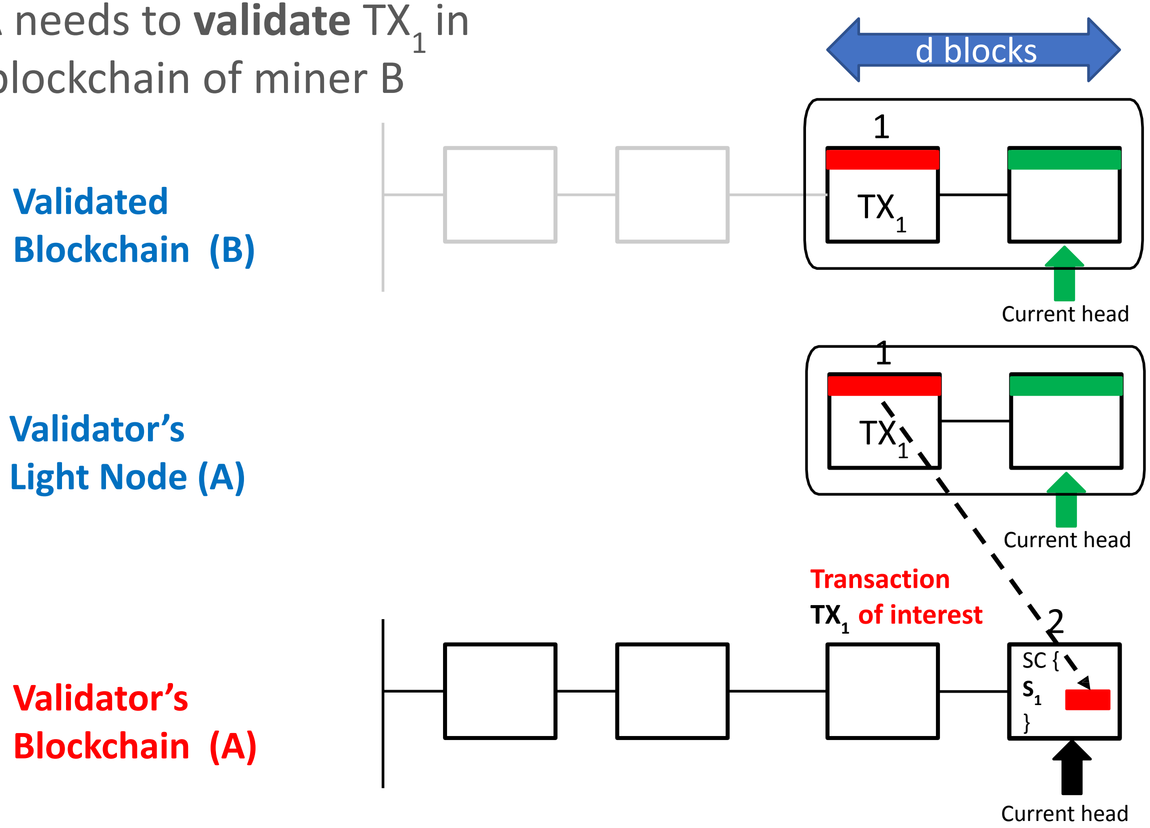
- Blockchain of 1 asset
- Light nodes of all other blockchains

Drawback : Does not scale well with large number of blockchains

Reason : Onus of cross-chain validation on one blockchain

Cross-Chain Validation of Transaction

Validator A needs to **validate** TX_1 in **validated** blockchain of miner B



A's light node stores last stable block of B

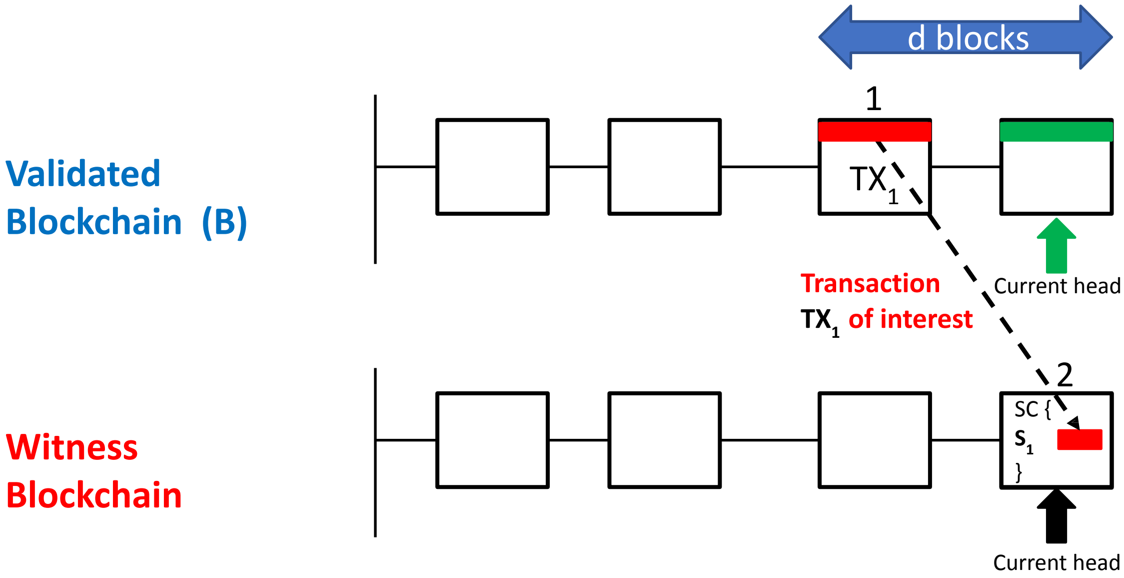
Cross-Chain Validation of Transaction

Proposed Method : Maintain a **witness blockchain**

- Each block stores smart contract containing header of **last stable block** of validated blockchain
- If last stable block of all blockchains commits transaction, then transaction is executed
- If last stable block of one blockchain aborts transaction, then transaction aborts
- Miner only requires to stores
  - Blockchain of 1 asset
  - Witness blockchain

Cross-Chain Validation of Transaction

Witness blockchain needs to **validate** TX_1 in **validated** blockchain of miner B



Witness blockchain validates TX_1 instead of regular blockchain

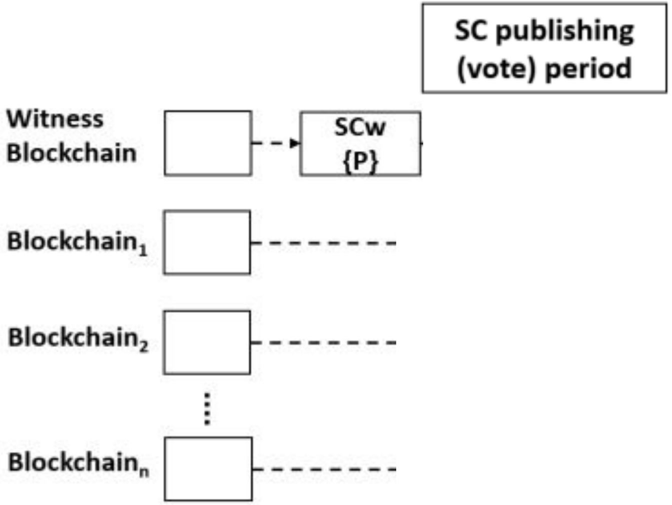
AC^{3}WN
Atomic Cross-Chain Commitment using Witness Network

- Use **witness blockchain** the atomic swap
- The witness blockchain decides the commit or the abort of a swap
  - All sub-transactions in the swap must follow the decision
  - Achieves atomicity, either **all committed** or **all aborted**
- The AC^{3}WN protocol has four phases:
  - Transaction deployment phase
  - Sub-transaction deployment phase
  - Witness network state change phase
  - Sub-transaction redemption/refund phase

AC^3WN
Atomic Cross-Chain Commitment using Witness Network

Transaction deployment phase

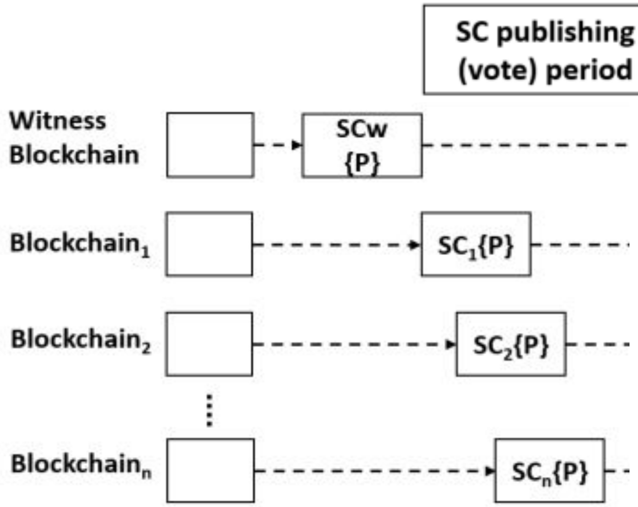
- Participant deploys contract SC_w in the witness network with state *Published (P)*
- SC_w has a header of a block at depth **d** of all blockchains in the swap



AC^3WN
Atomic Cross-Chain Commitment using Witness Network

Sub-transaction parallel deployment phase

- Participants deploy their contracts in their corresponding asset blockchains

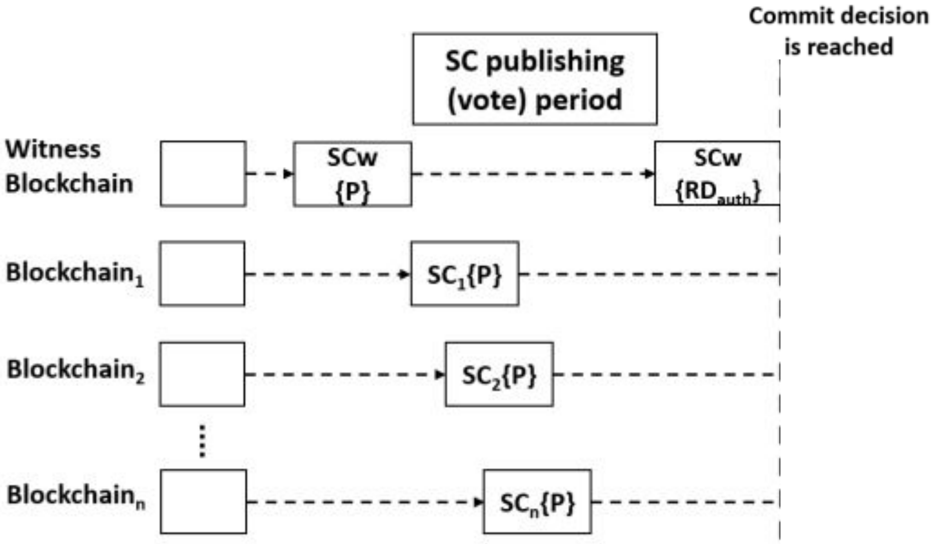


AC^3WN

Atomic Cross-Chain Commitment using Witness Network

Witness network state change phase

- Participants submit evidence of publishing smart contracts in the asset blockchain
- If all contracts are published and correct, SC_w's state is altered to redeem (RD)

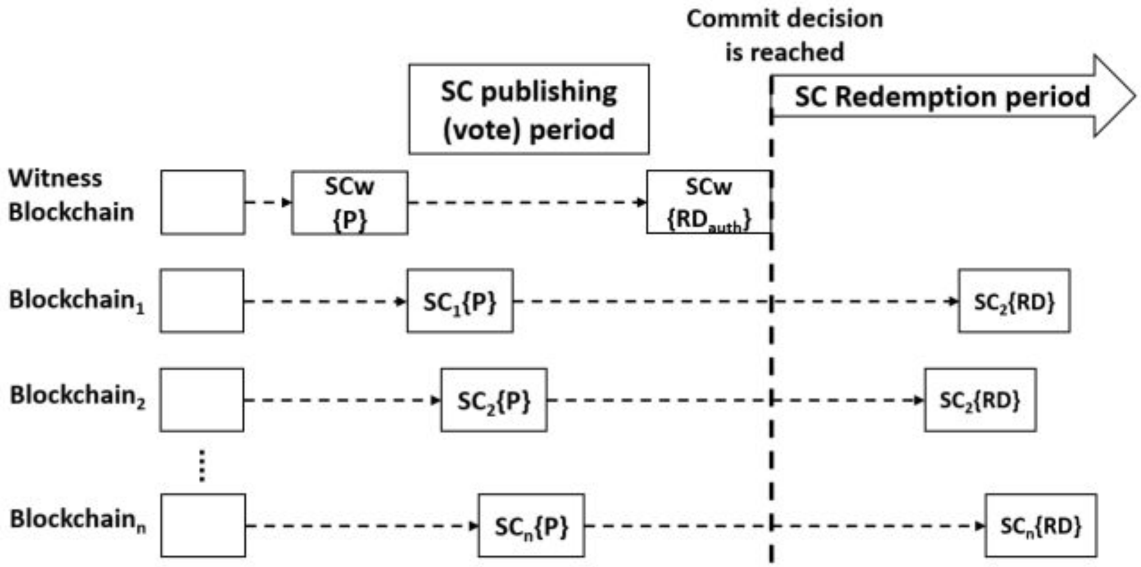


AC^3WN

Atomic Cross-Chain Commitment using Witness Network

Sub-transaction redemption/refund phase

- Participants submit evidence of redeem state (RD) from witness blockchain to asset blockchains.
- After evidence verification, participants redeem their assets from asset blockchains.



AC^3WN
Atomic Cross-Chain Commitment using Witness Network

Conditions to redeem :

- No participant backs out of transaction while SC_w is in P
- All participants deploy SCs for their sub-transactions to respective blockchains
- No malicious activity
  - Individual blockchain validates sub-transactions
  - SC_w can validate all blocks after the last stable blocks in each blockchain

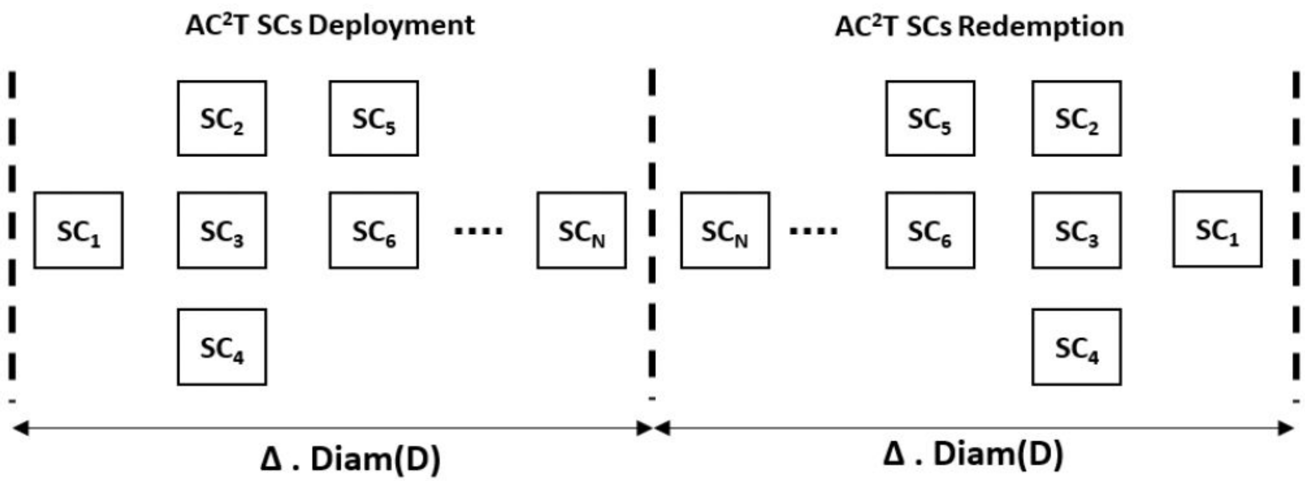
If any condition fails, then the transaction is **refunded**.

Advantages of AC^3WN

- Transaction doesn’t depend on rational behaviour of participants
  - Only requires network of witnesses to validate transaction
- Participants cannot violate atomicity
  - Commit and abort decided only by witness blockchain
- Honest participants can concurrently publish SCs without worrying about malicious participants
- Works on asynchronous environment
  - Out-of-sync participants use SC_w as evidence for redemption/refunding of asset

Evaluation

- **Latency :** Difference between time t_s when transaction starts, and time t_c when it is completed.
- Δ **:** Time for a participant to either :
  - Publish SC in blockchain
  - Change SC state through a function call of this SC, and for this change to be publicly recognized.
- Consider AC^{2}T to be a directed graph **D = (V,E)**.
- **Diam(D) :** Length of longest path between any 2 vertices in D.
- A single leader swap protocol (Nolan/Herlihy) has 2 phases
  - Deployment phase which has a latency of Δ **. Diam(D)**.
  - Redemption phase which has a latency of Δ **. Diam(D)**.

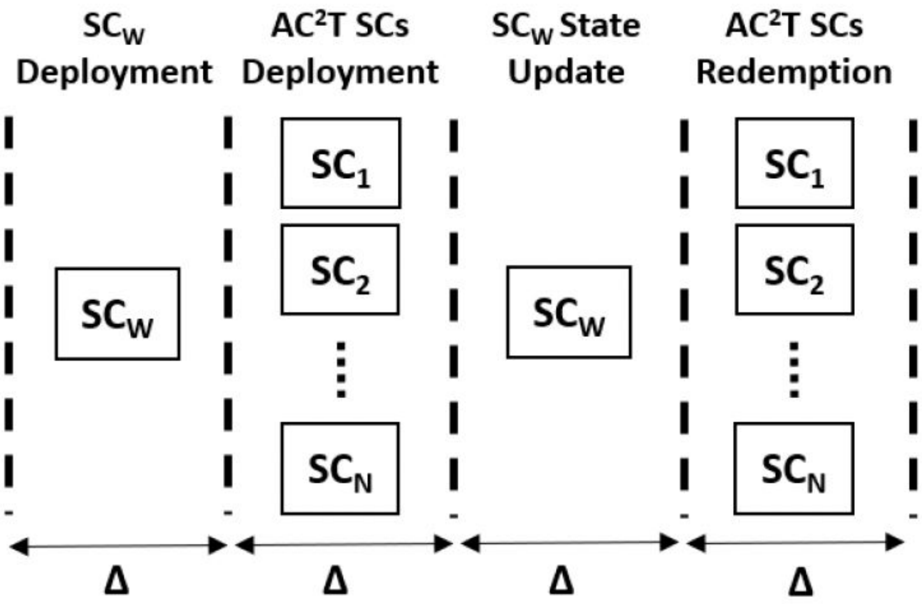


The overall transaction latency of $2 \cdot \Delta \cdot Diam(\mathcal{D})$

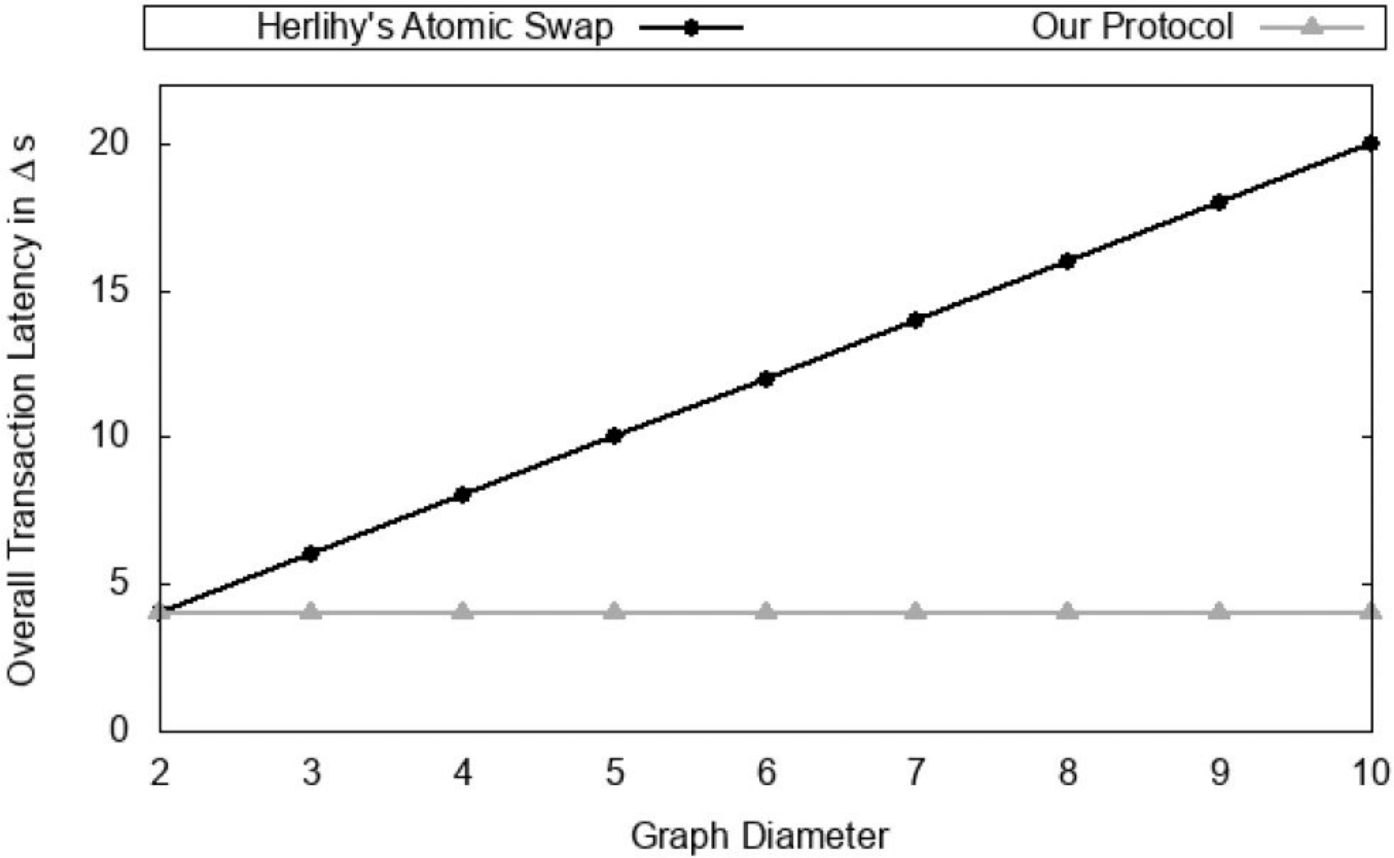
Evaluation

The AC^{3}WN protocol has four phases:

- The witness network SC deployment phase - Δ
- The transaction SC parallel deployment phase - Δ
- The witness network SC state change phase - Δ
- The transaction SC parallel redemption phase - Δ



The overall transaction latency of $4 \cdot \Delta$


Herlihy's Atomic Swap
Our Protocol
Overall Transaction Latency in Δ s
0
5
10
15
20
2
3
4
5
6
7
8
9
10
Graph Diameter

Cost Overhead

- Both protocols
  - Deploy a smart contract for every edge.
  - Invoke redemption or refund function call for every deployed smart contract in the AC^2T
  - Results in N function calls.
- AC^3WN
  - Requires deployment of additional contract SC_w in witness network
  - Additional function call to change state of SC_w either from P to RD or from P to RF.
- AC^2T fee
  - Herlihy's protocol : $N * (f_d + f_{fc})$
  - AC^3WN protocol : $(N + 1) * (f_d + f_{fc})$.

Conclusion

- First decentralized cross-chain transaction protocol ensuring all-or-nothing atomicity
  - Even during participant crash failures and network DoS attacks.
- AC^3WN separates the coordination of an AC^2T from its execution
  - Permissionless open network of witnesses coordinate AC^2T
  - Participants in the AC^2T execute sub-transactions in the AC^2T.
- This separation enables AC^3WN to parallely execute sub-transactions in the AC^2T
  - Reduces the latency of AC^2T.

Thank You!