CONSENSYS

Crosschain Protocol Stack & DeFi Bridge Design

Peter Robinson, November 23, 2021

Agenda

- Introduction to DeFi Bridges
- Crosschain Protocol Stack
- Design Choices

CONSENSYS

Introduction to DeFi Bridges

Bridge?

Bridges allow you to:

- Swap assets on one blockchain for assets on another blockchain.
  Transfer assets from one blockchain to another blockchain.
  - Assets are typically ERC 20 fungible tokens, ERC 721 non-fungible tokens.
- Communicate arbitrary data and messages across blockchains.
- Have functions in contracts on one blockchain call functions in contracts on another blockchain.

A ***Bridge*** is also known as a:

- Blockchain Bridge
- Crosschain Bridge
- DeFi Bridge
- Token Bridge

Why?

Why would you want to have a bridge from blockchain to another?

- Transaction fees:
    - Different blockchains have different fees. These are typically driven by the number of people competing to use a blockchain. Note that lower transaction fees might come with more centralization and lower security.
- Block confirmation times:
    - Typically, you should wait between six and twelve block confirmations* on Ethereum MainNet.This translates to between 1 ½ and 3 minutes. Other blockchains may offer faster block times and faster or instant finality.
      Note 1: that faster confirmation times may come with more centralization and lower security.
      Note 2: the Ethereum Merge will change the block confirmation times.
- Functionality:
    - A contract you wish to use may be hosted on a different blockchain to the one you have your value on.

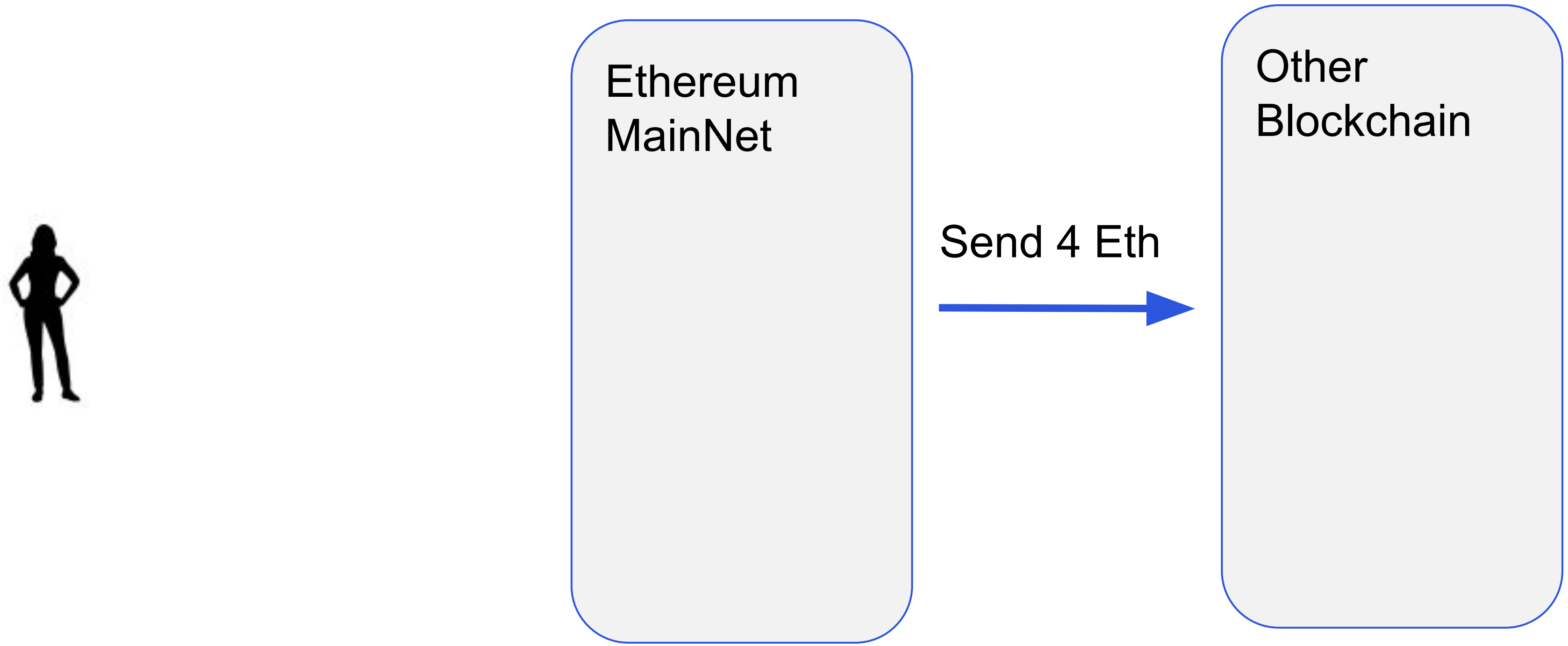
* See https://doi.org/10.1017/S0269888920000296 or open access version: https://arxiv.org/abs/1906.04421

Why?

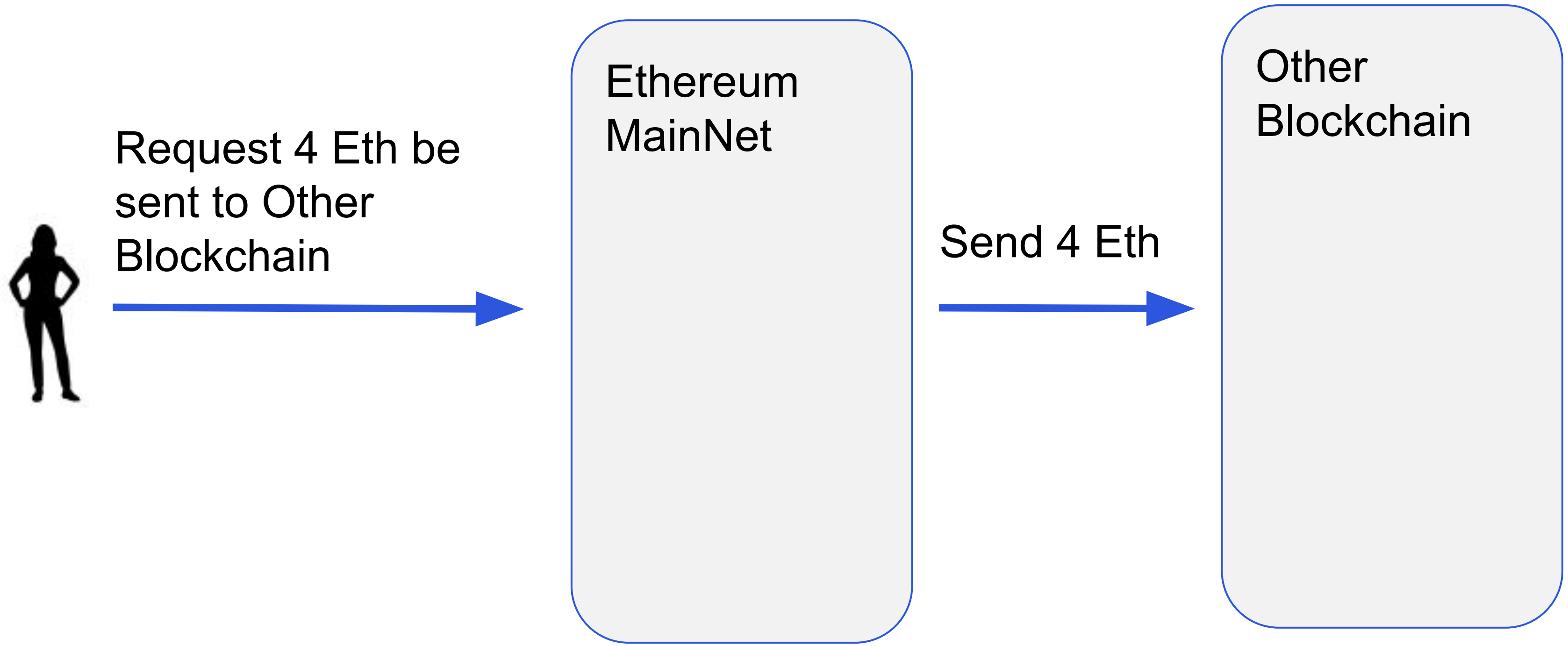
Why would you want to have a bridge from blockchain to another?

- Liquidity pools:
  - Moving your value to a blockchain with a larger liquidity pool may make it easier to trade.
- Capital utility / efficiency
  - Able to use all assets from all blockchains in the one transaction.

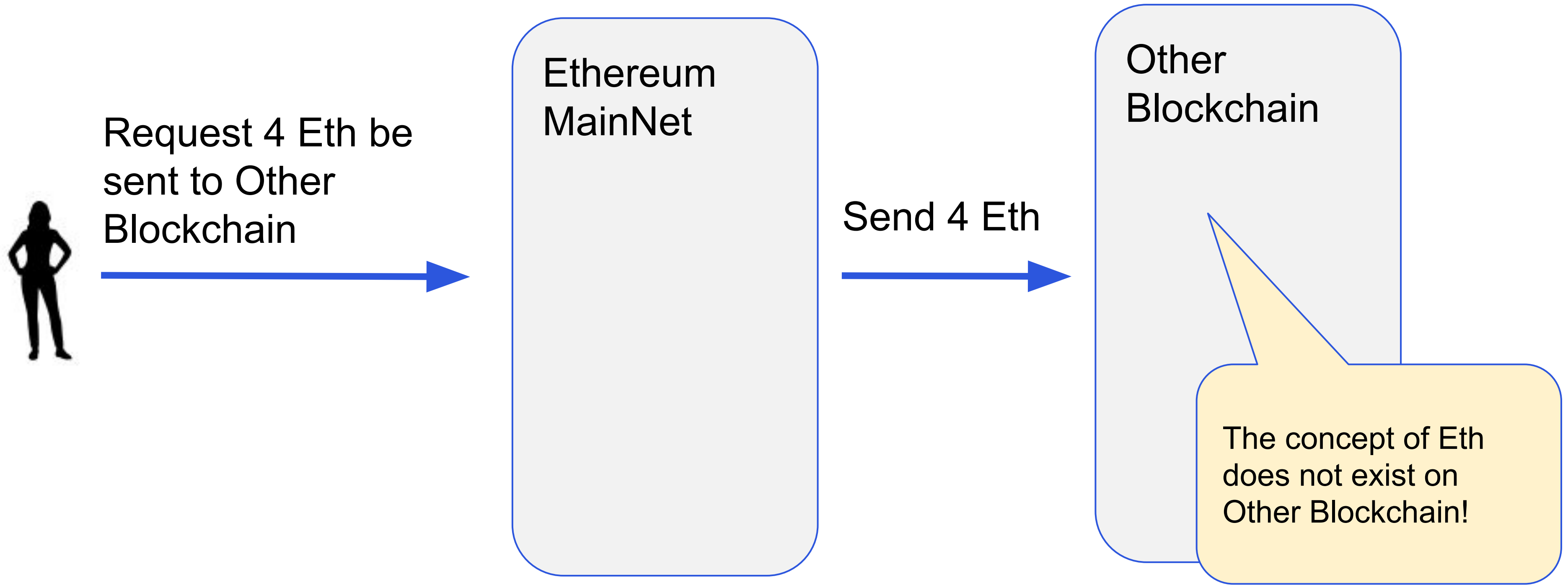
Value Transfer



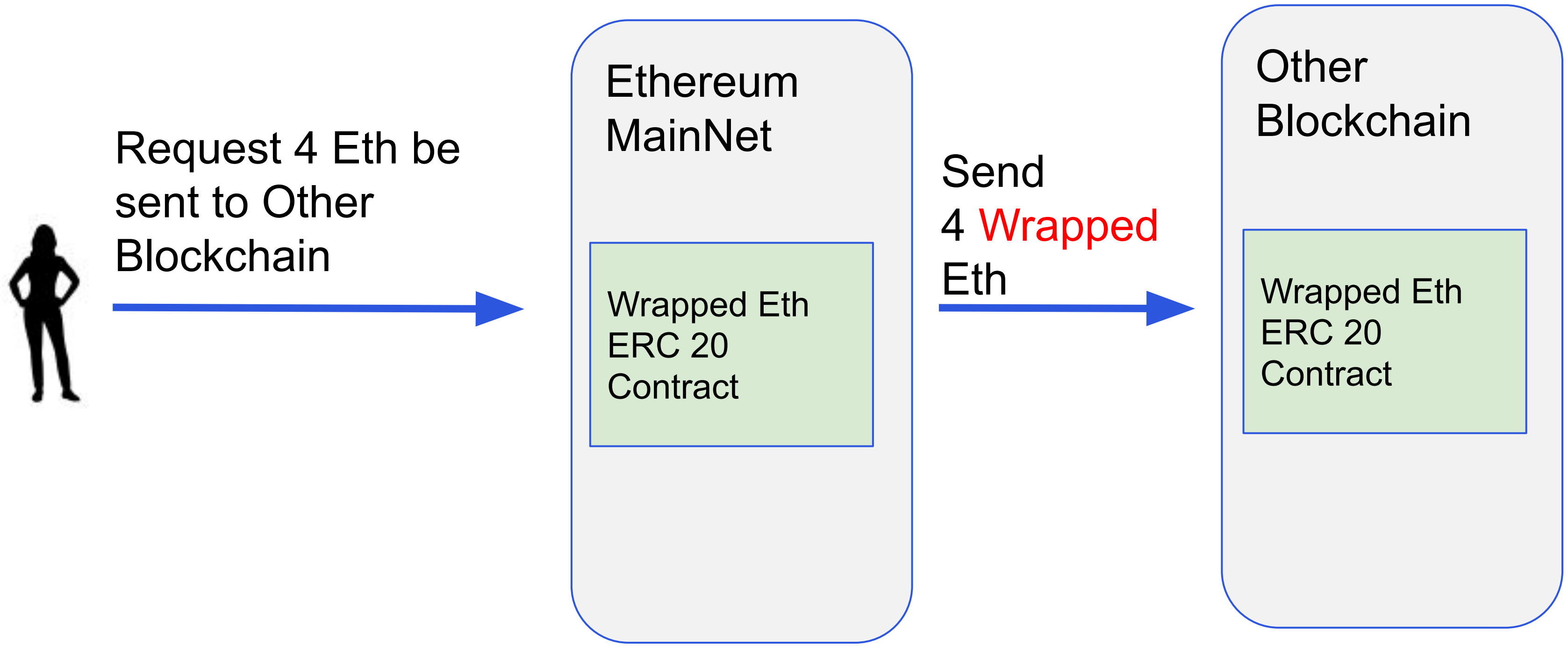
Value Transfer



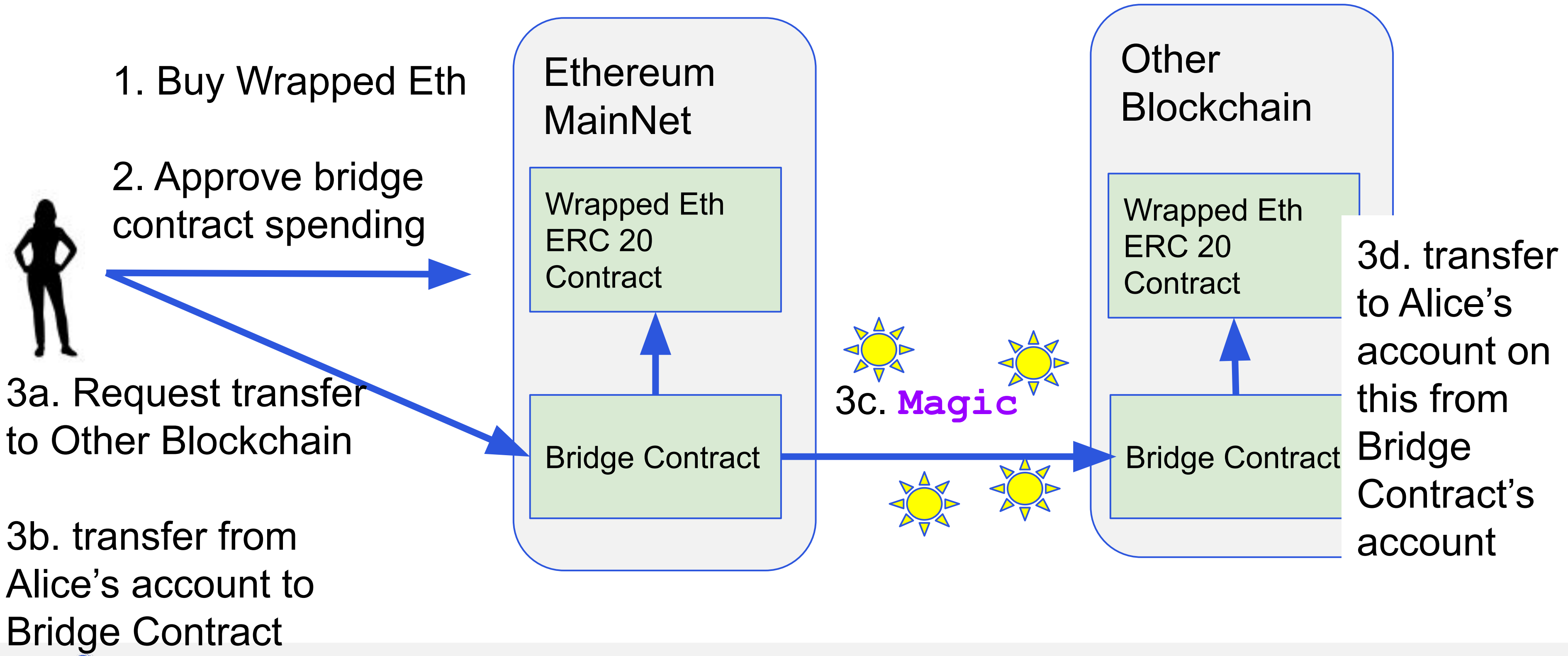
Value Transfer


Request 4 Eth be sent to Other Blockchain
Ethereum MainNet
Send 4 Eth
Other Blockchain
The concept of Eth does not exist on Other Blockchain!

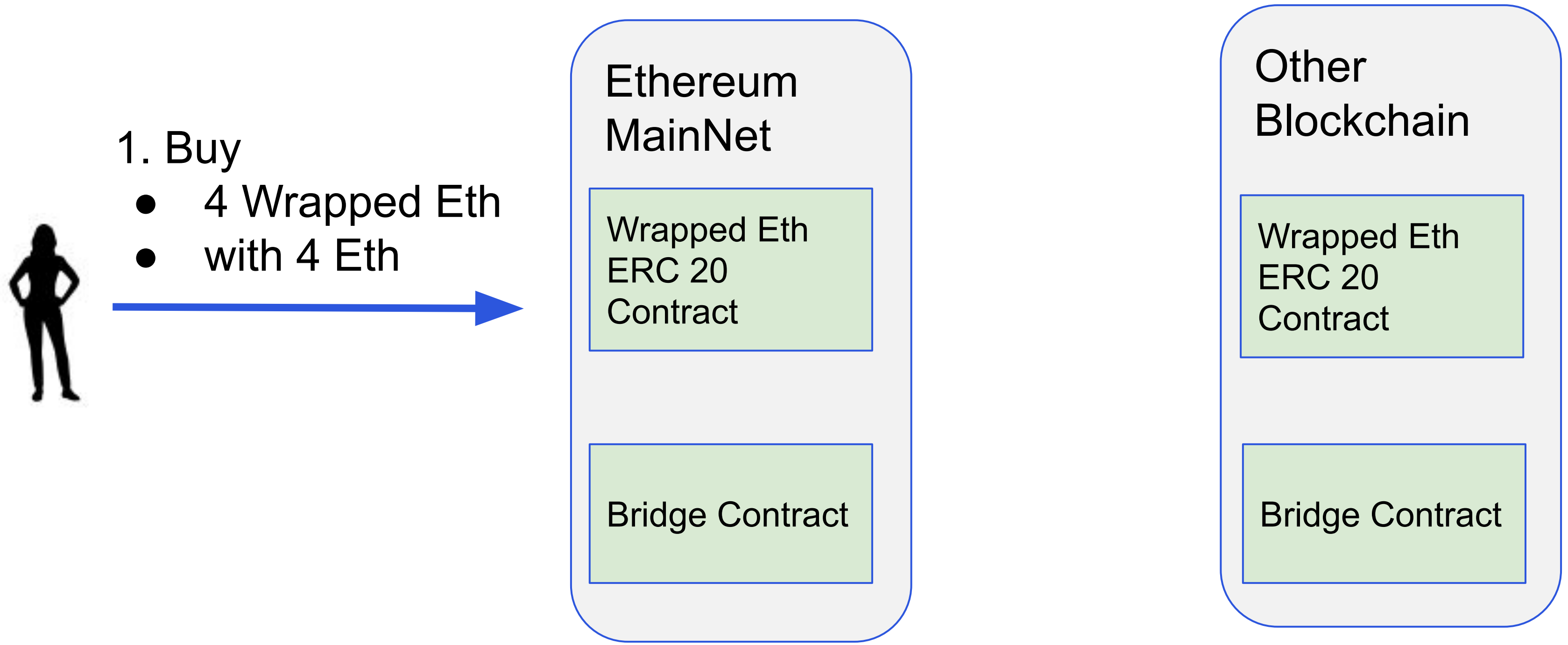
Value Transfer


Request 4 Eth be sent to Other Blockchain
Ethereum MainNet
Wrapped Eth ERC 20 Contract
Send 4 Wrapped Eth
Other Blockchain
Wrapped Eth ERC 20 Contract

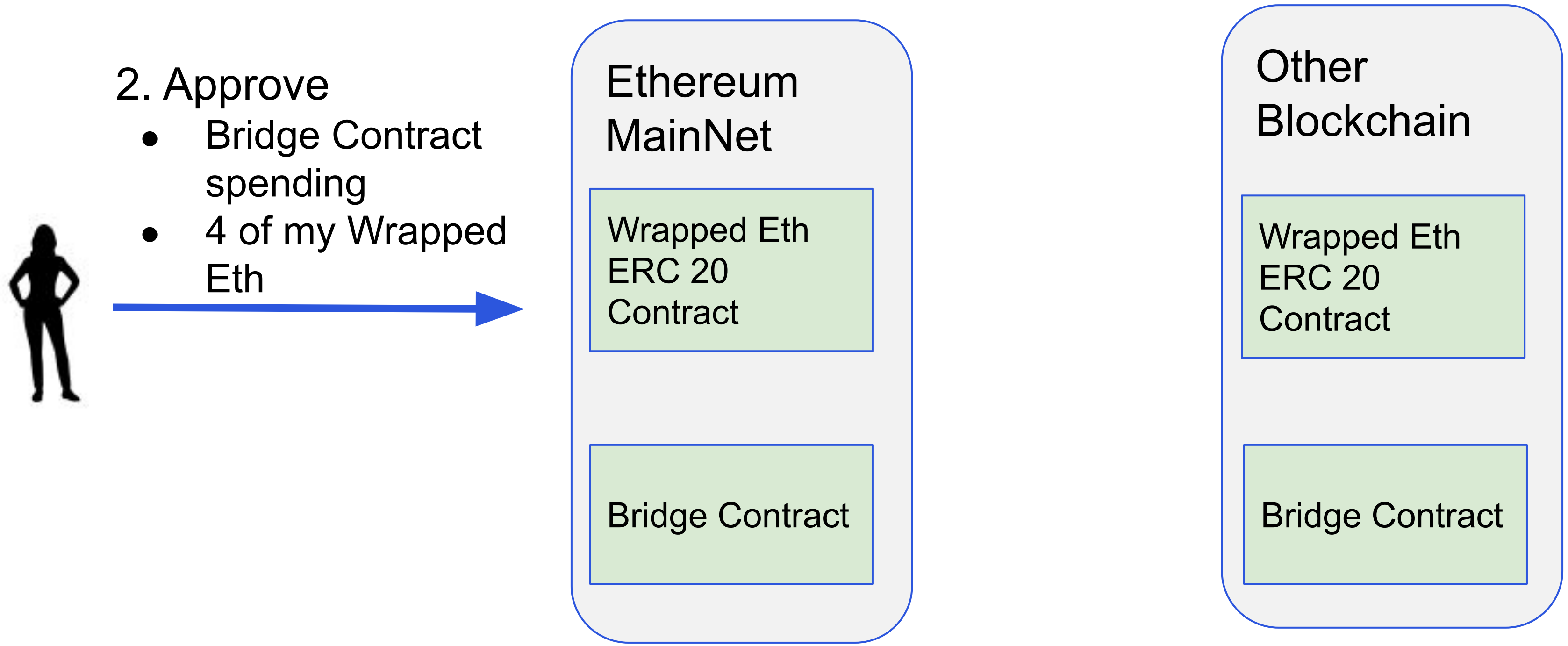
Value Transfer


1. Buy Wrapped Eth
2. Approve bridge contract spending
3a. Request transfer to Other Blockchain
3b. transfer from Alice's account to Bridge Contract
Ethereum MainNet
Wrapped Eth ERC 20 Contract
Bridge Contract
3c. Magic
Other Blockchain
Wrapped Eth ERC 20 Contract
Bridge Contract
3d. transfer to Alice's account on this from Bridge Contract's account

Value Transfer



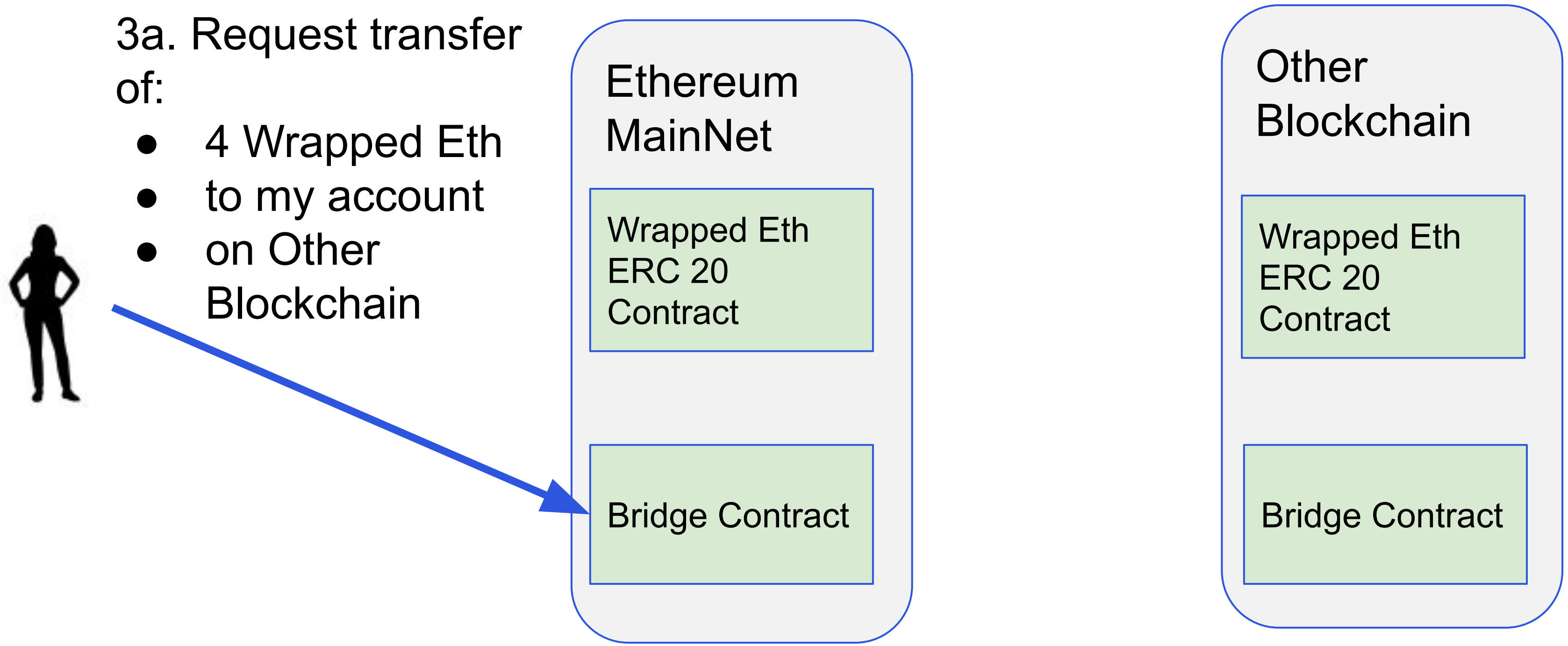
Value Transfer



Value Transfer

3a. Request transfer of:

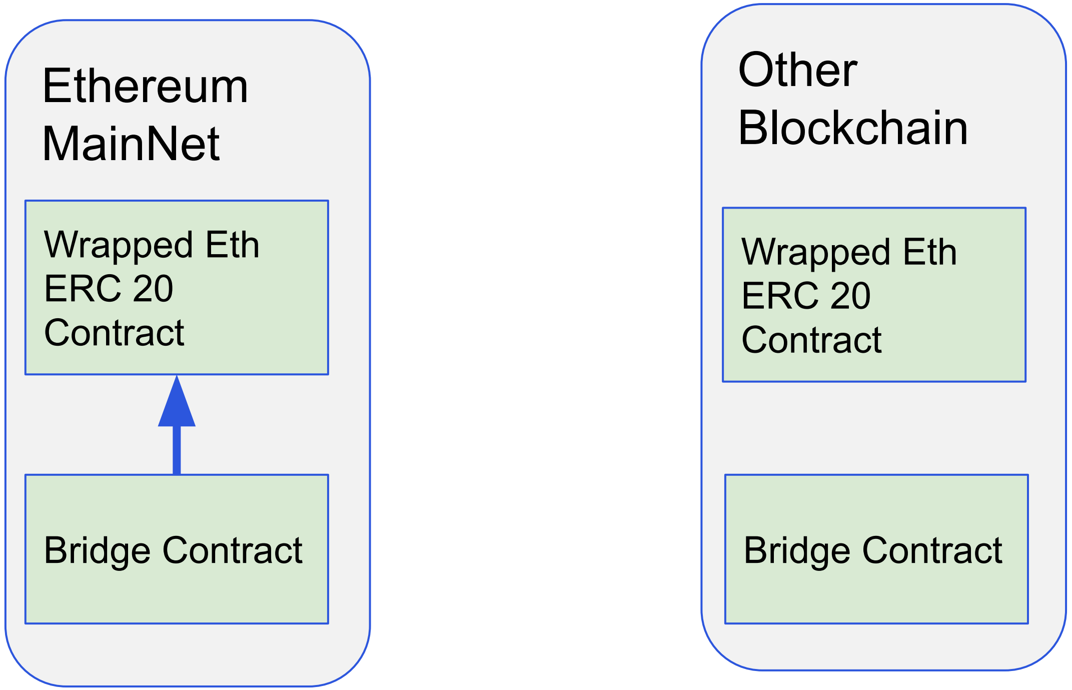
- 4 Wrapped Eth
- to my account
- on Other Blockchain



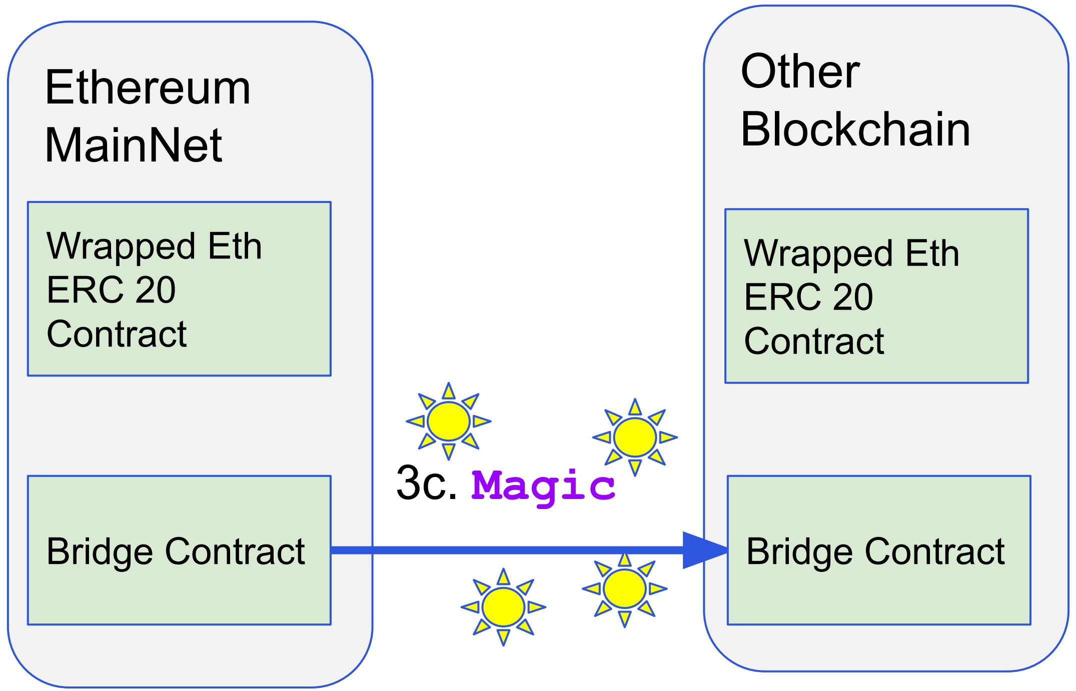
Value Transfer

3b. transfer from:

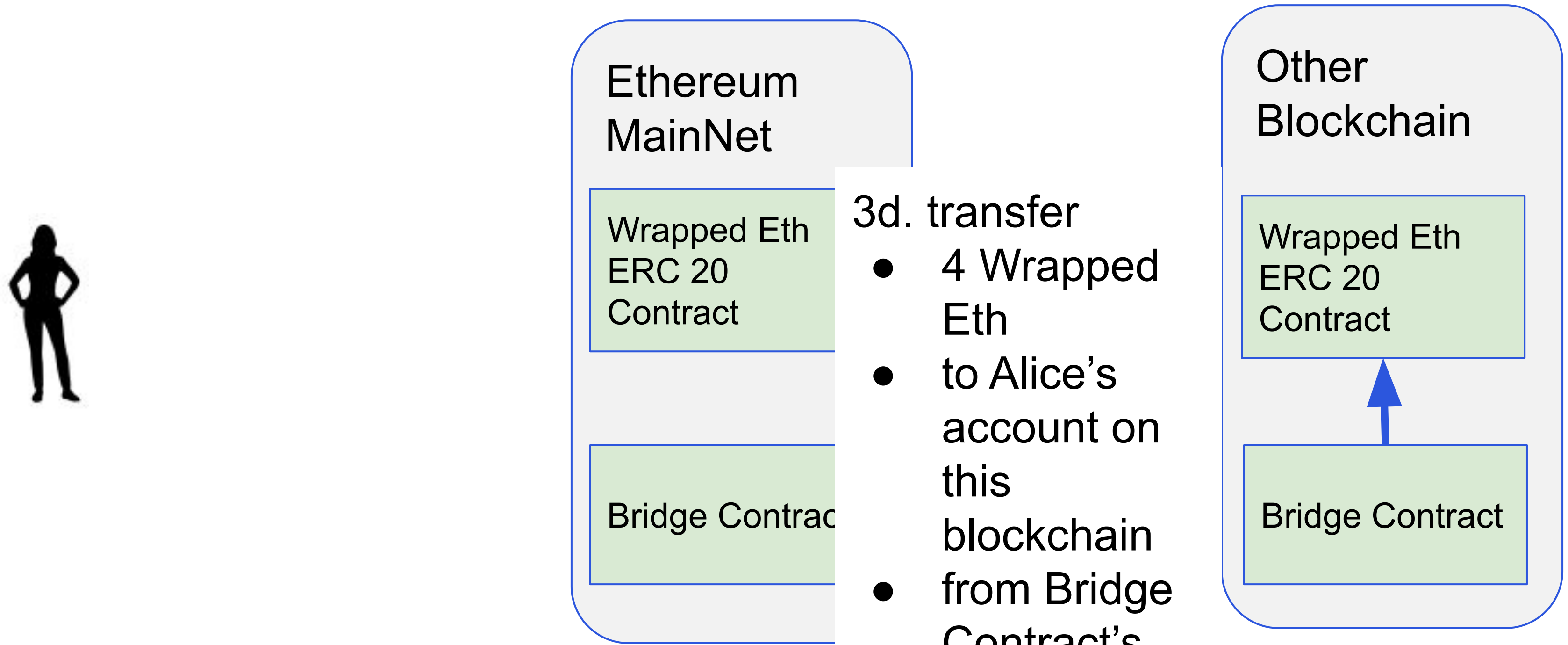
- Alice’s account
- to Bridge Contract account
- 4 Wrapped Eth



Value Transfer

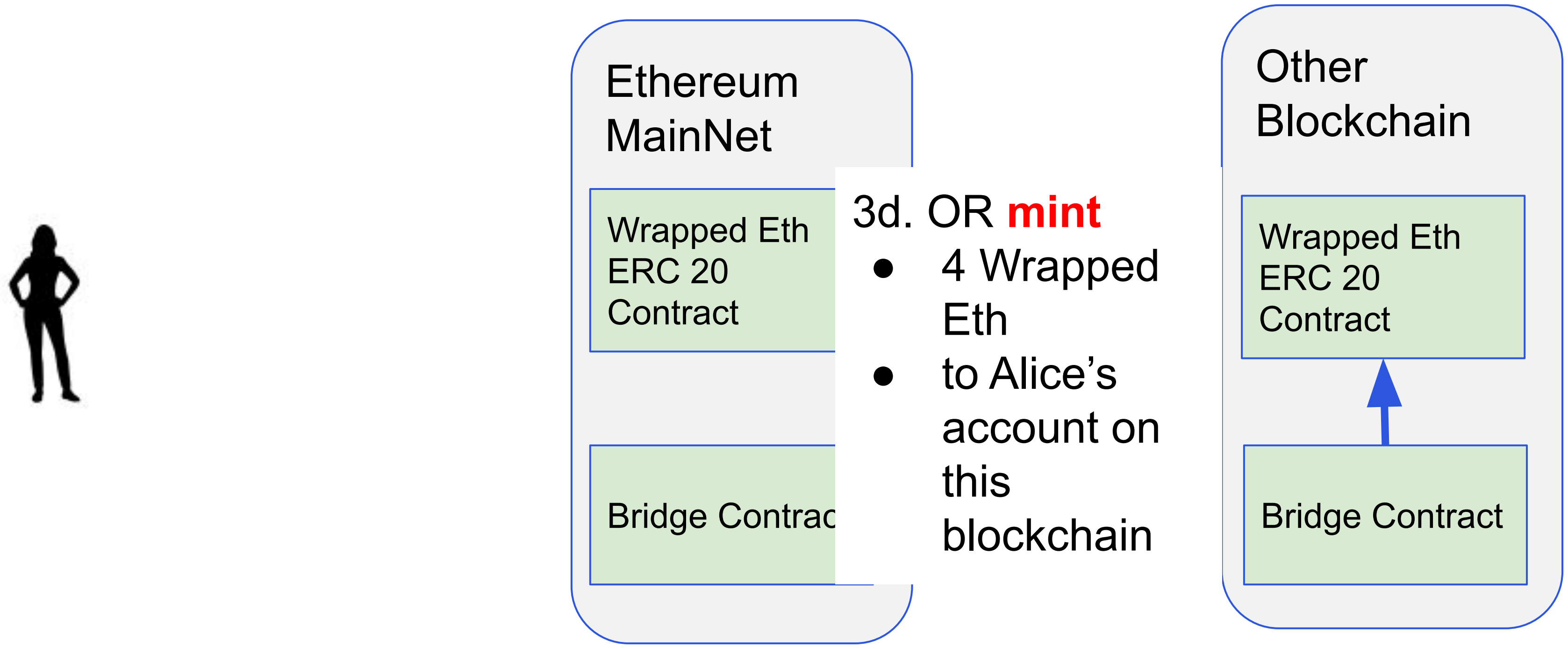


Value Transfer



3d. transfer
- 4 Wrapped Eth
- to Alice's account on this blockchain
- from Bridge Contract's account

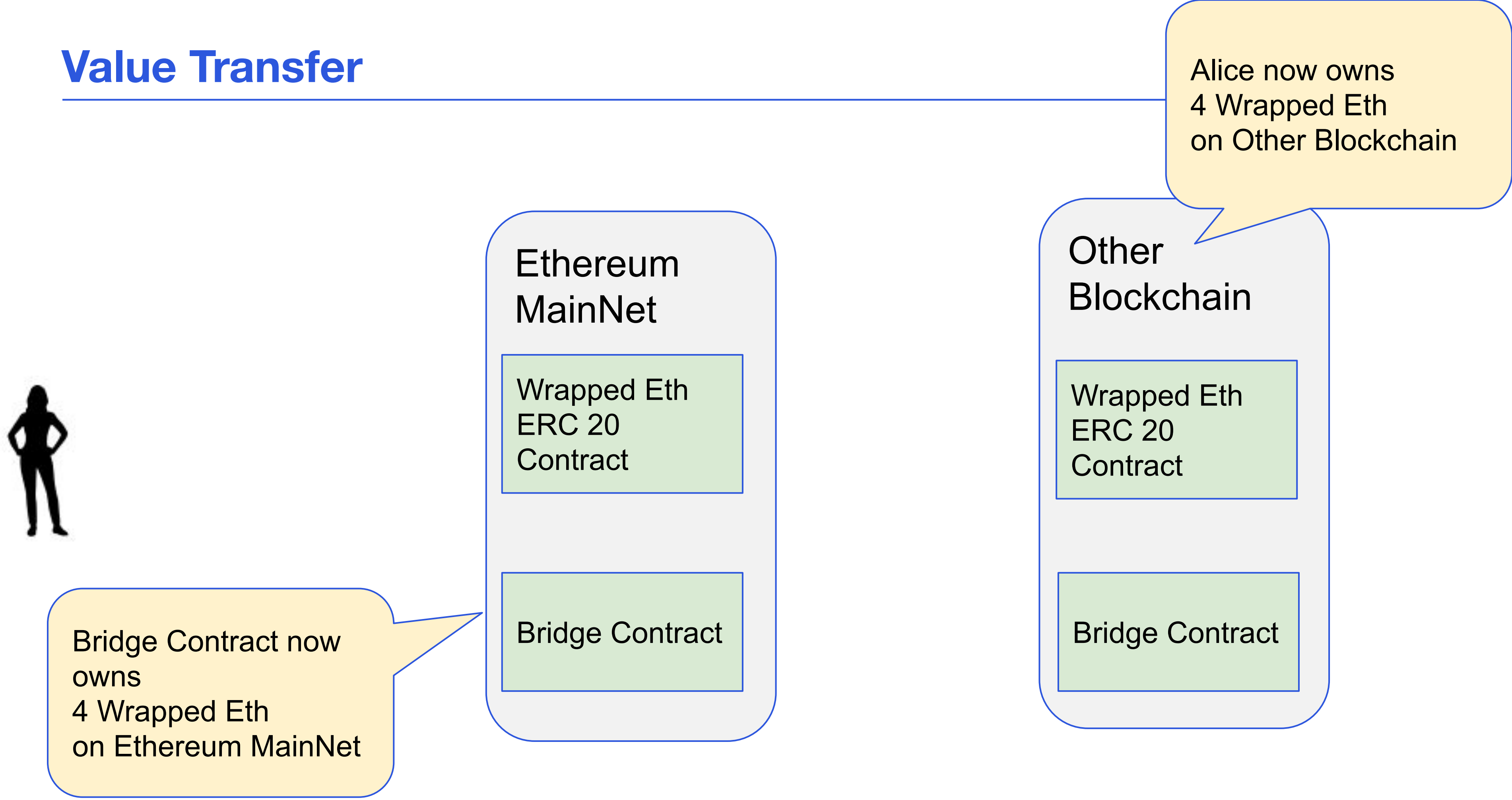
Value Transfer



3d. OR **mint**

- 4 Wrapped Eth
- to Alice's account on this blockchain

Value Transfer



Crosschain Use-Cases

Lots of other crosschain use-cases:

- Supply chain financing.
- Supply chain provenance / selective transparency.
- Cross-border supply chain.
- Inter-CBDC payments.

EEA: Crosschain Interoperability Working Group

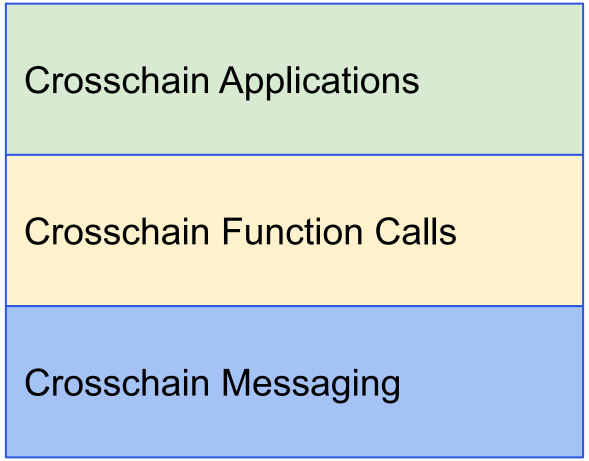
Crosschain Interoperability
Use Case

https://entethalliance.org/wp-content/uploads/2020/08/CIFT_Use_Case.pdf

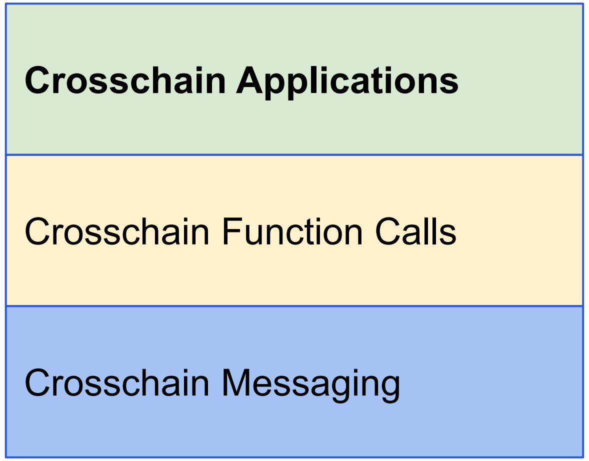
CONSENSYS

Crosschain Protocol Stack

Crosschain Protocol Stack



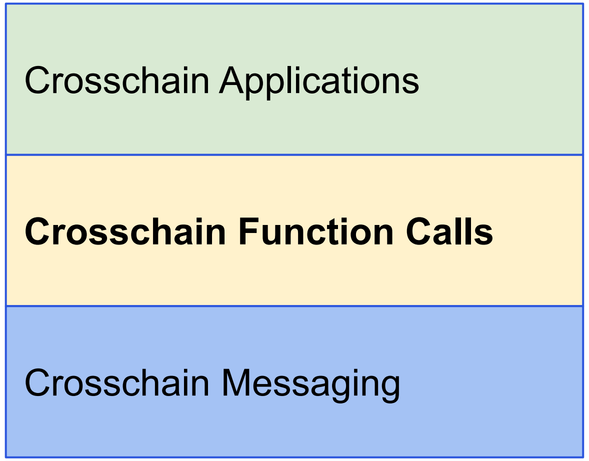
Crosschain Protocol Stack



The **Crosschain Applications** layer consists of applications that operate across blockchains.

There may be software components created in this layer that allow complex applications to be created more easily.

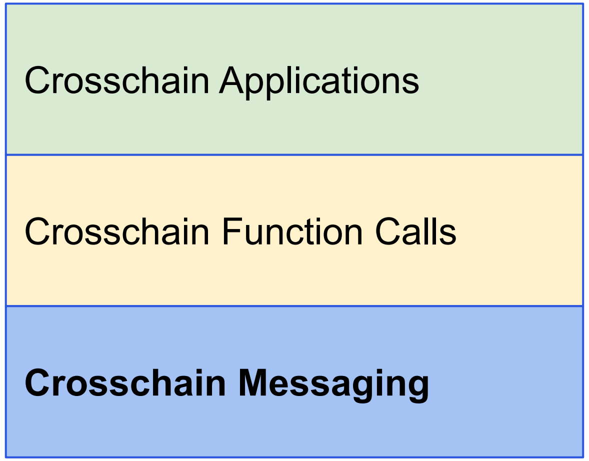
Crosschain Protocol Stack



The **Crosschain Function Calls** layer executes function calls across blockchains.

The updates due to the function call protocol can be atomic or not-atomic.

Crosschain Protocol Stack



The **Crosschain Messaging** layer ensures that information can be verified as having come from a certain blockchain.

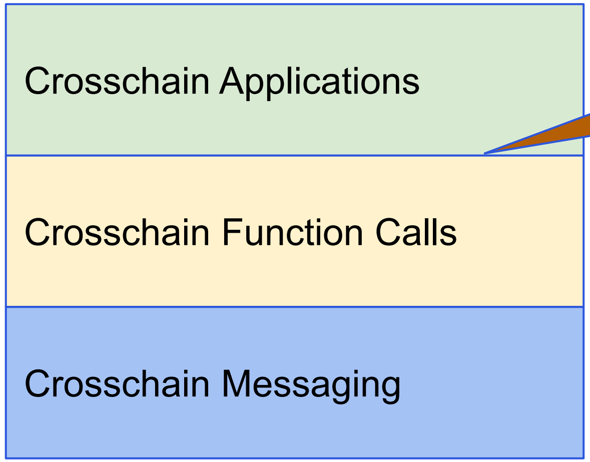
Crosschain Protocol Stack

Having a layered architecture with **clearly defined interfaces between layers** has the following advantages:

- **Interoperability**: Applications will work with a variety of Function Call implementations that will work with a variety of Messaging implementations.
- **Flexibility**: different Crosschain Message Verification technique could be used for each blockchain / roll-up* used in an overall crosschain function call.
- **Infrastructure Reuse**: Different Crosschain Function Call techniques can share the same deployed Crosschain Message Verification infrastructure.
- **Focus on what you are good at**: Organisations can focus on creating solutions for a single part of the protocol stack stack. That is, rather than having to create the entire stack, a company might choose to focus on an application, a better Function Call approach, or some Messaging infrastructure.
- **Experimentation**: Not having to build the entire protocol stack should make experimentation much easier.

* Rollups: https://www.youtube.com/watch?v=Ohi6R7m5BWk&t=4s

Crosschain Protocol Stack



Interfaces between layers being standardised in the Enterprise Ethereum Alliance's Crosschain Interoperability Working Group.

CONSENSYS

Design Choices: Crosschain Messaging Layer

Crosschain Messaging: Ethereum Events

Ethereum transactions can emit *events*:

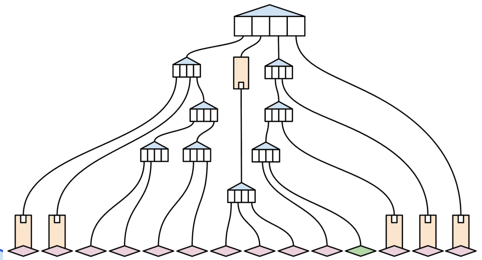
```
function transfer(address _recipient, uint256 _amount) external {
  ....

  emit Deposit(msg.sender, _recipient, _amount);
}

event Deposit(address _from, address _to, uint256 _amount);
```

Crosschain Messaging: Ethereum Events

- Events are stored in transaction receipts.
- The root of a Merkle Patricia Trie of transaction receipts is included in the Ethereum Block Header.

Block 100

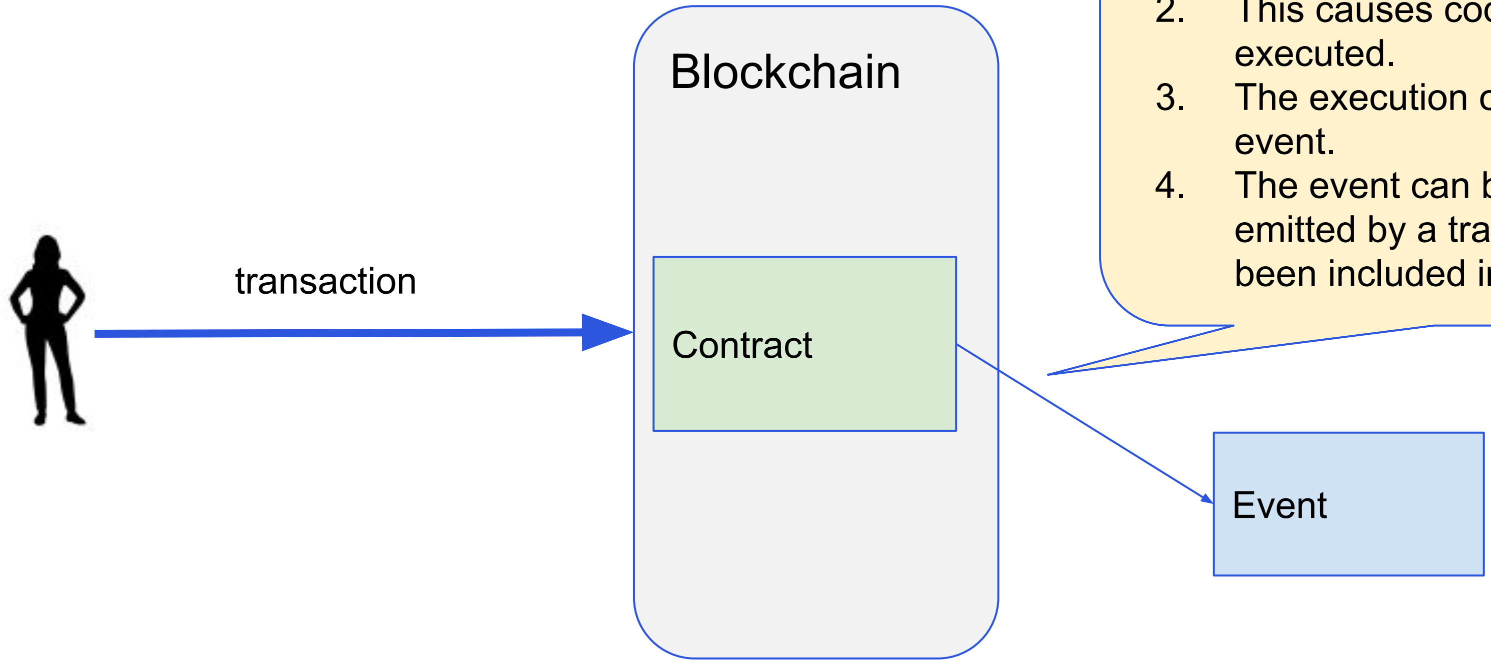
Block 101:
… , transaction receipt root, ...



Transaction Receipt:
- State root or status,
- Cumulative gas used,
- Bloom Filter,
- Event logs,
- Revert Reason

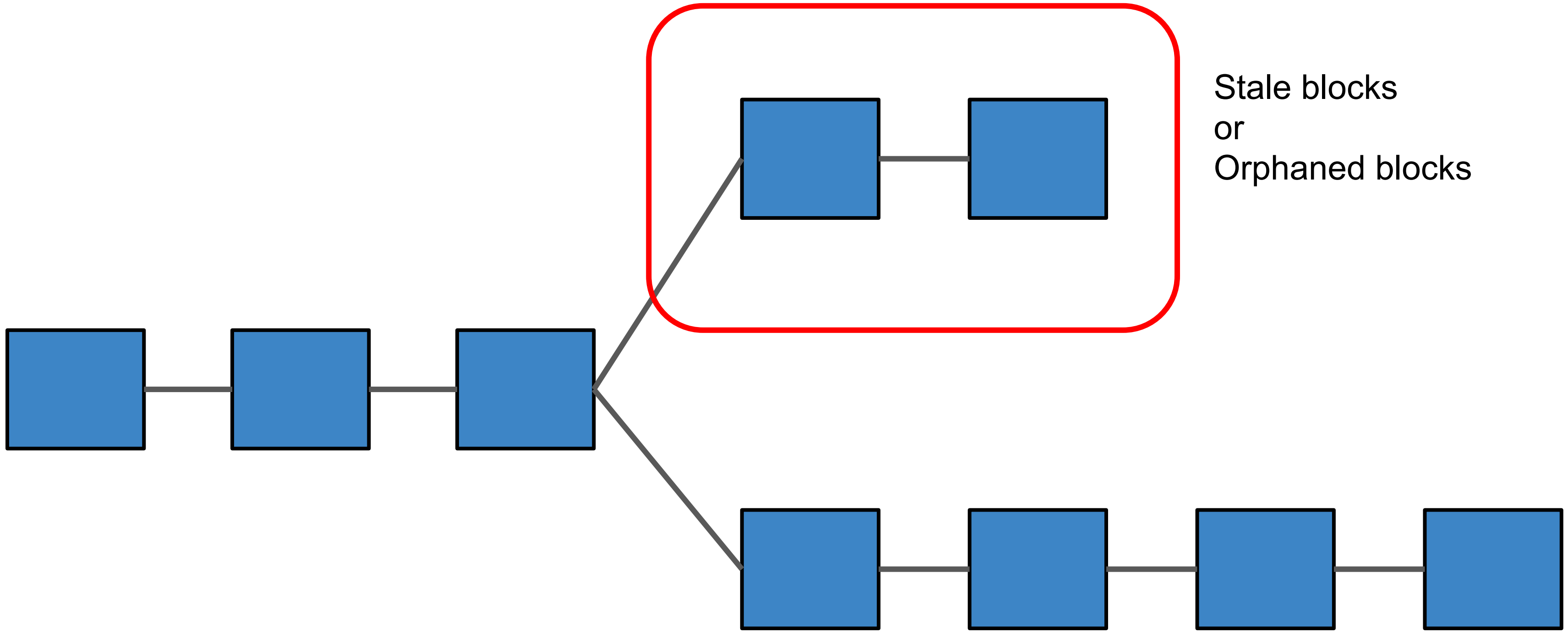
Event log:
- Address of contract that emitted the event,
- Topics. These are the event function signature and indexed parameters.
- Data. The encoded parameters.

Crosschain Messaging: Ethereum Events

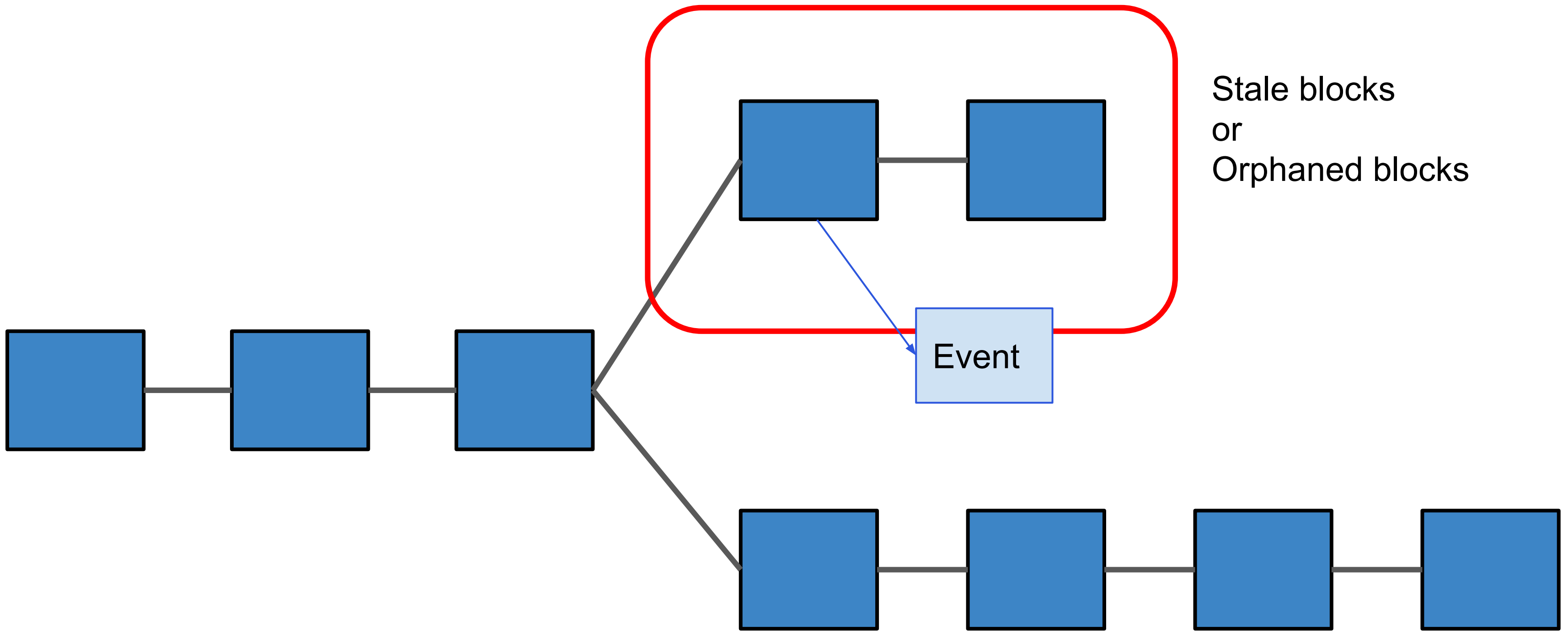


1. The user submits a transaction.
2. This causes code in the contract to be executed.
3. The execution of the code emits an event.
4. The event can be proven to be emitted by a transaction that has been included in a block.

Crosschain Messaging: Finality



Crosschain Messaging: Finality



Stale blocks
or
Orphaned blocks

Crosschain Messaging: Finality

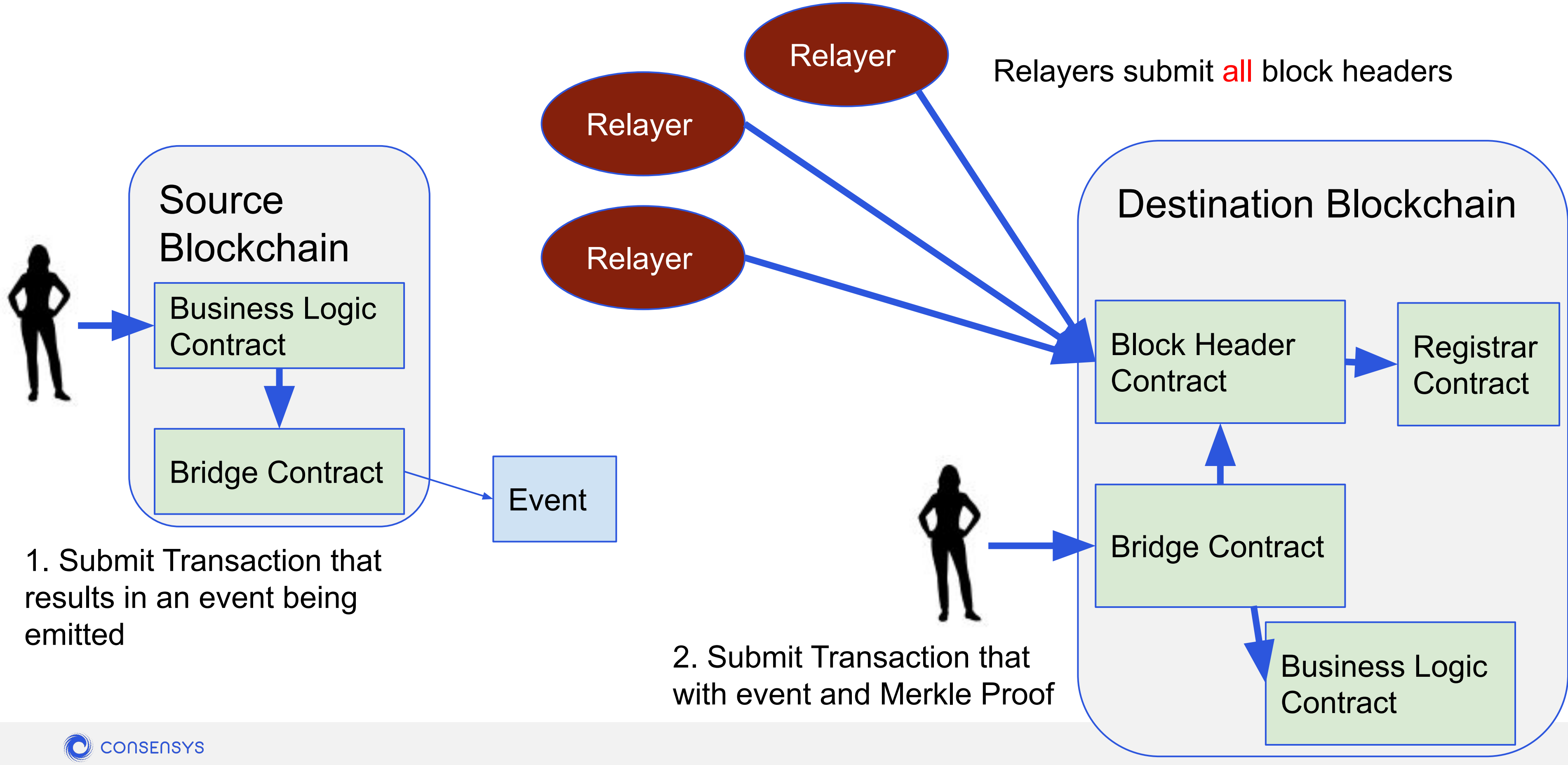
- **A Crosschain Messaging system should only use events that belong to transactions that have been included in blocks that are (probably) final.**

- Block confirmation times:
    - Typically, you should wait between six and twelve block confirmations* on Ethereum MainNet.This translates to between 1 ½ and 3 minutes.
    - Other blockchains may offer faster block times and faster or instant finality.
    - Note that faster confirmation times may come with more centralization and lower security.
    - Note: the Ethereum Merge will change the block confirmation times.

* See https://doi.org/10.1017/S0269888920000296 or open access version: https://arxiv.org/abs/1906.04421

Block header transfer, separately submit header

This is approximately how Clearmatics' Ion Framework works



Block header transfer, separately submit header

This is approximately how Clearmatics' Ion Framework works

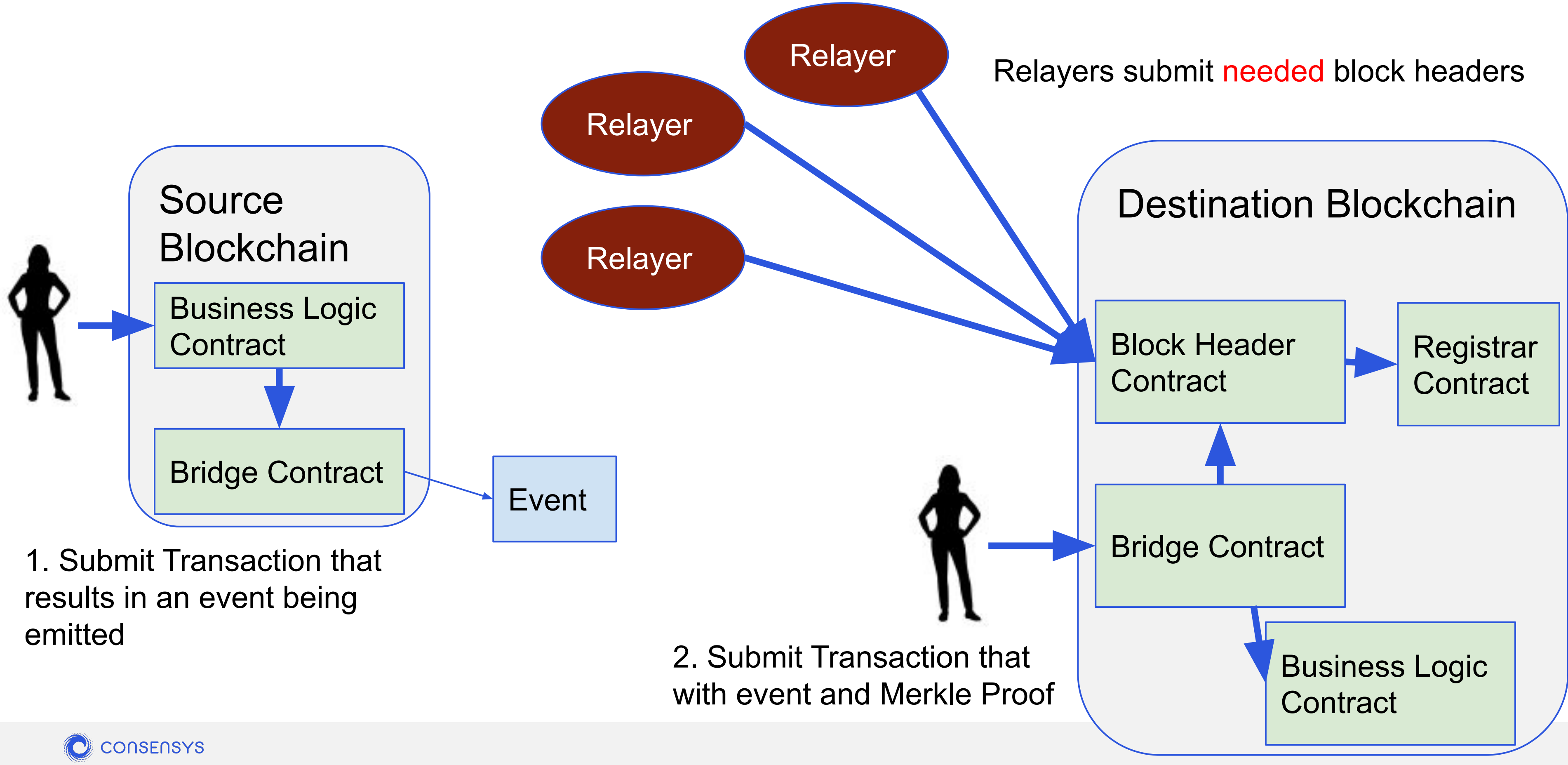
Advantage:

- Simple.
- Relayers do not need to cooperate.
- Users do not interact with Relayers.

Disadvantage:

- One transaction for each relayer, for each block to submit signed block headers!
- If there are multiple target blockchains, this process has to be repeated for all target blockchains.
- Relayers are paying to submit transactions on the destination blockchain.
- User has to be able to submit a transaction on the destination blockchain.

Block header transfer, each separately submit block header



Block header transfer, each separately submit block header

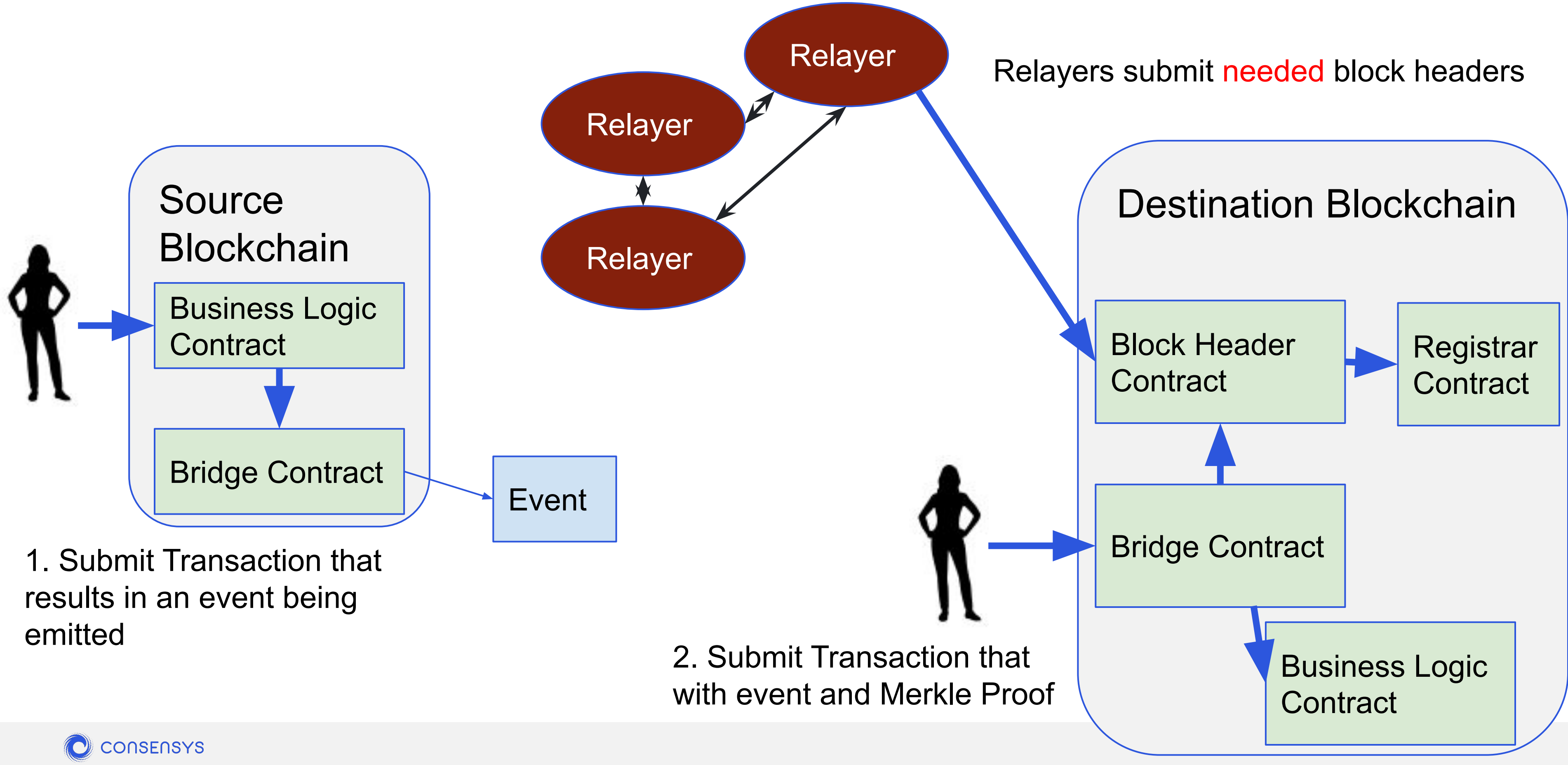
Advantage:

- One transaction per Relayer per block header that is needed.
- Relayers do not need to cooperate.
- Users do not interact with Relayers.

Disadvantage:

- Relayers need to analyse transactions in blocks to determine which block headers need to be transferred.
- If there are multiple target blockchains, relayers need to understand which blockchain an event will be used on, OR block headers that might be needed need to be communicated with all blockchains.
- Relayers are paying to submit transactions on the destination blockchain.
- User has to be able to submit a transaction on the destination blockchain.

Block header transfer, Relayers cooperate to multiply sign



Block header transfer, Relayers cooperate to multiply sign

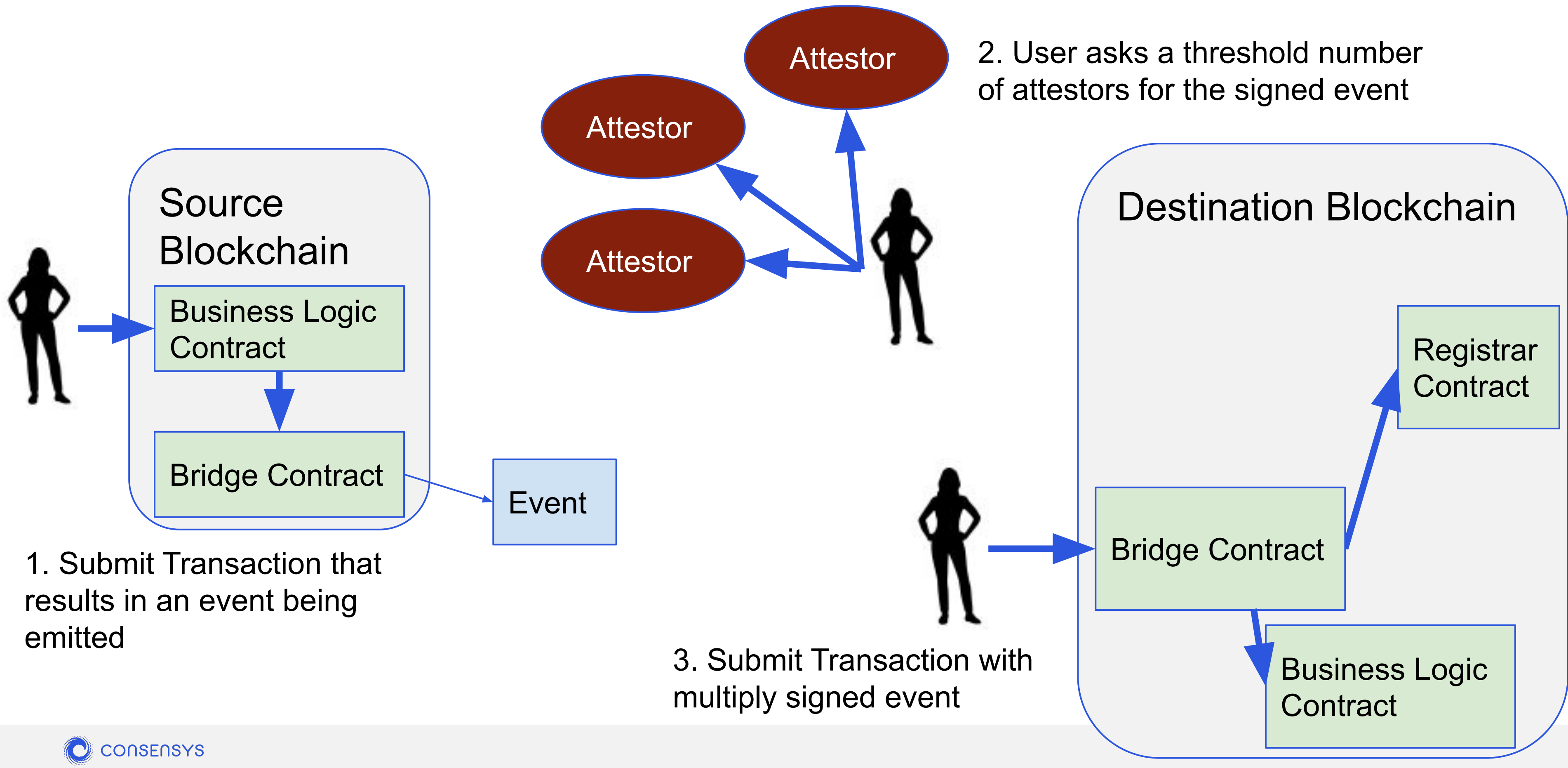
Advantage:

- One transaction per block header that is needed.
- Users do not interact with Relayers.

Disadvantage:

- Relayers need to cooperate.
- Relayers need to analyse transactions in blocks to determine which block headers need to be transferred.
- If there are multiple target blockchains, relayers need to understand which blockchain an event will be used on, OR block headers that might be needed need to be communicated with all blockchains.
- Relayers are paying to submit transactions on the destination blockchain.
- User has to be able to submit a transaction on the destination blockchain.

Event sign, separate Attestor sign



2. User asks a threshold number of attestors for the signed event

1. Submit Transaction that results in an event being emitted

3. Submit Transaction with multiply signed event

Event sign, separate Attestor sign

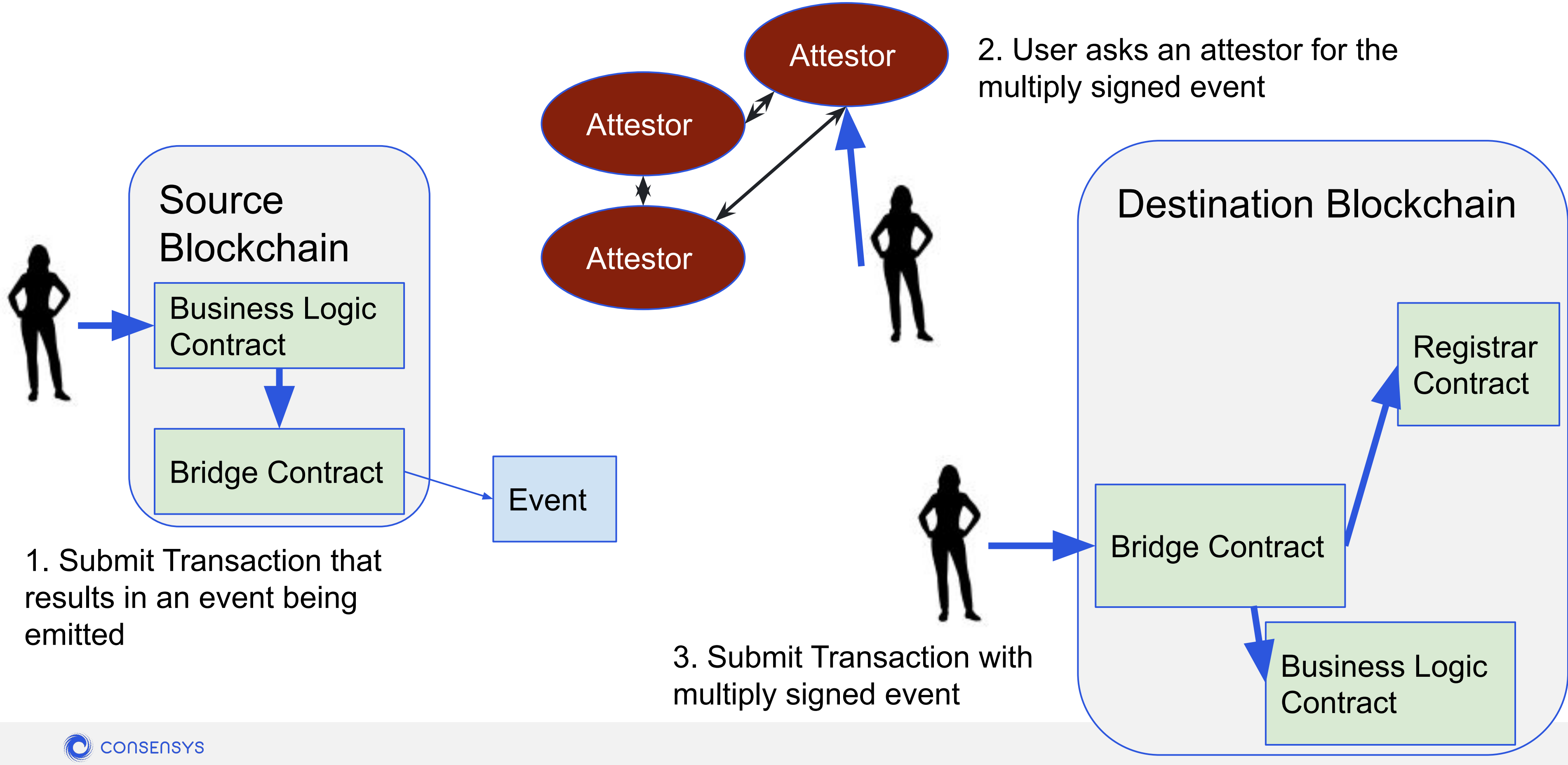
Advantage:

- Attestors do not submit transactions on the destination blockchain.
- Attestors sign all transactions from a certain contract. They don’t need to know understand the target blockchain.
- Attestors do not need to cooperate.

Disadvantage:

- Users interact with Attestors.
- User has to be able to submit a transaction on the destination blockchain.

Event sign, Attestors cooperate to multiply sign



1. Submit Transaction that results in an event being emitted

2. User asks an attestor for the multiply signed event

3. Submit Transaction with multiply signed event

Event sign, Attestors cooperate to multiply sign

Advantage:

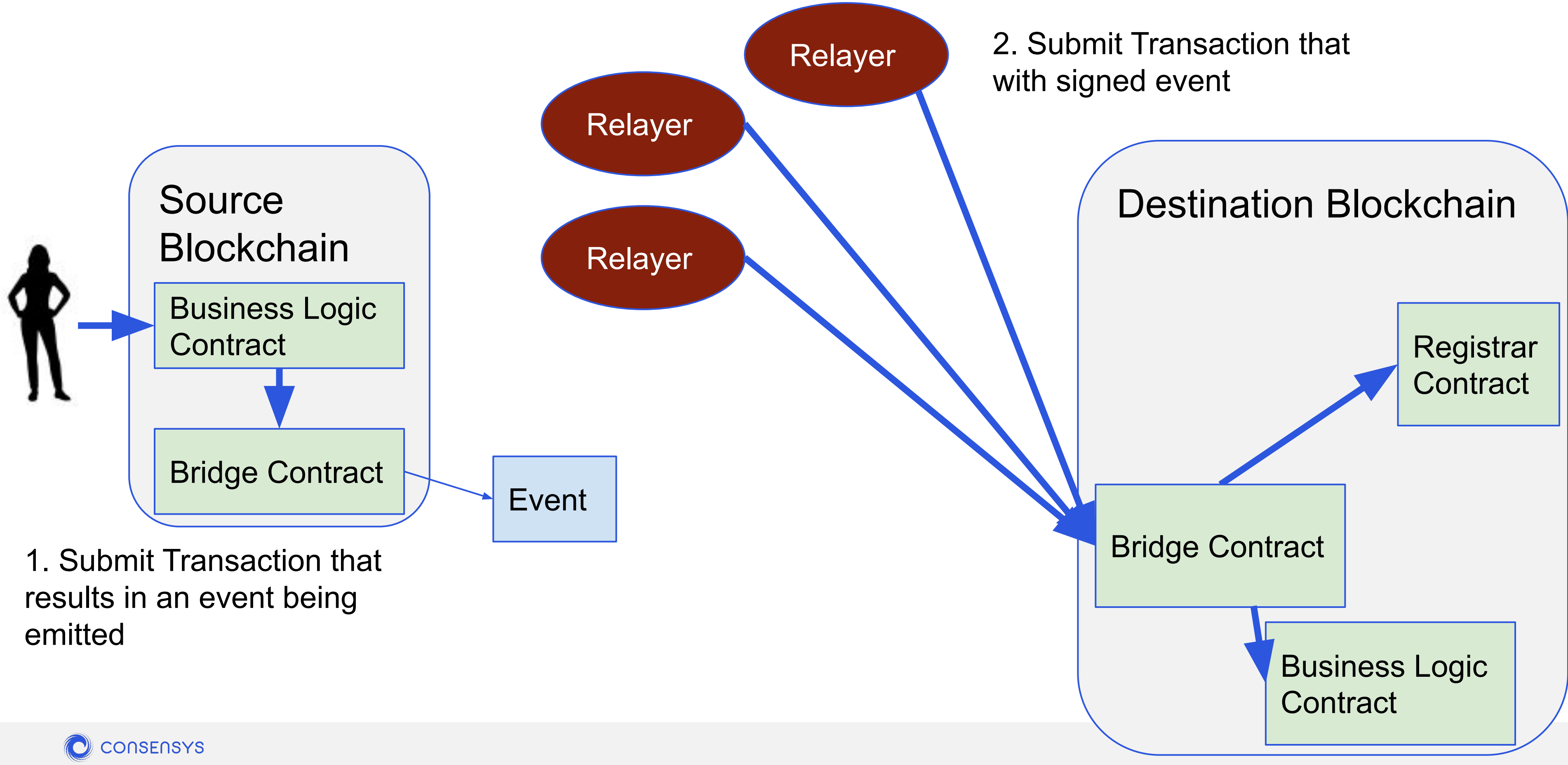
- Attestors do not submit transactions on the destination blockchain.
- Attestors sign all transactions from a certain contract. They don't need to know understand the target blockchain.

Disadvantage:

- Attestors need to cooperate.
- Users interact with Attestors.
- User has to be able to submit a transaction on the destination blockchain.

This is approximately how ChainSafe's ChainBridge works

Relayer separately submit signed event



Relayer separately submit signed event

Advantage:

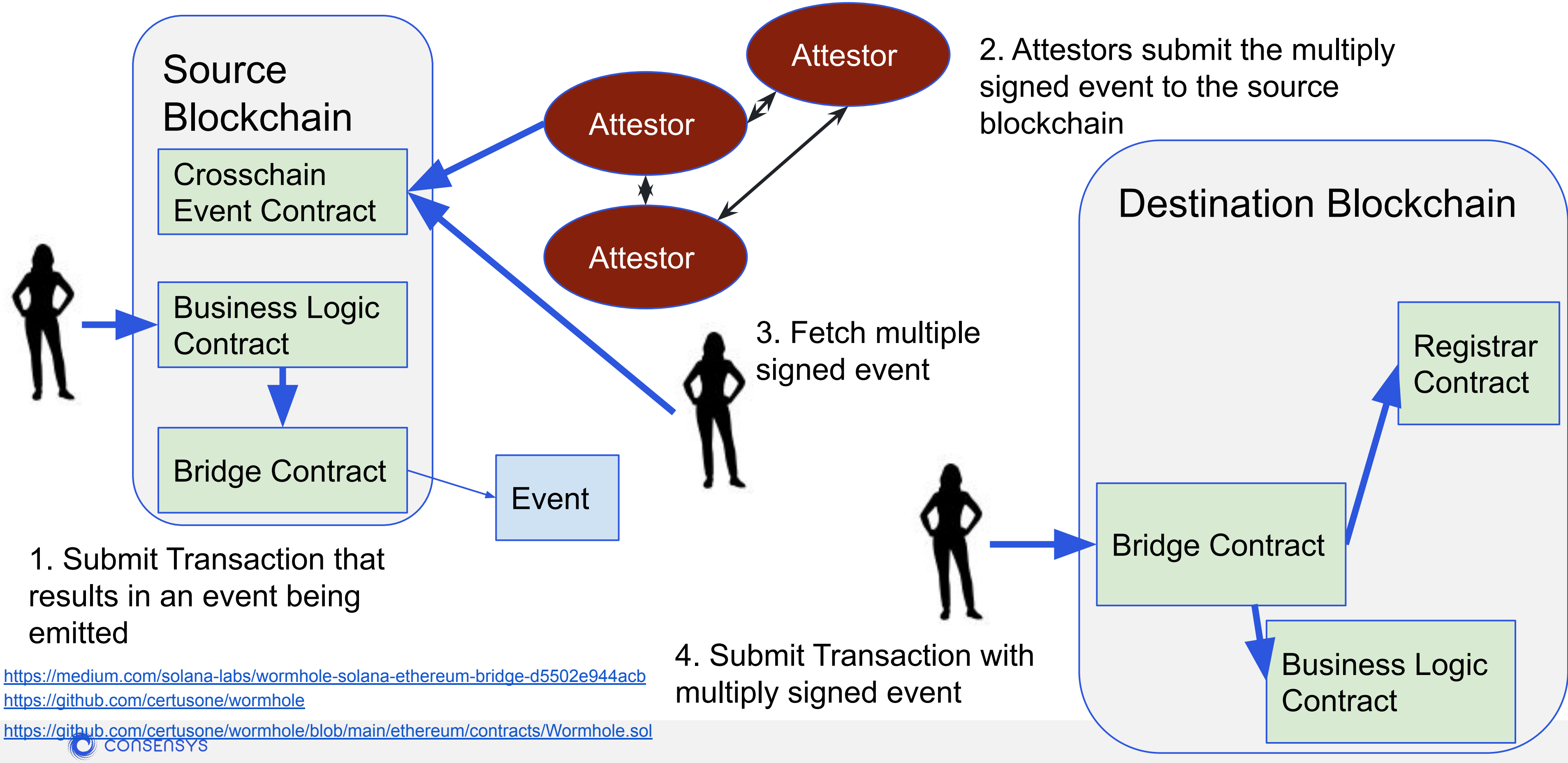
- Relayers do not need to cooperate.
- Users do not interact with Relayers.
- Users do not need to be able to submit transactions on the destination blockchain.

Disadvantage:

- One transaction per Relayer per event.
- Relayers submit transactions on the destination blockchain.
- Relayers need to know understand the target blockchain for an event.

Attestors submit multiply signed event to source

This is approximately how the Wormhole Solana - Ethereum MainNet bridge works



2. Attestors submit the multiply signed event to the source blockchain

3. Fetch multiple signed event

1. Submit Transaction that results in an event being emitted

4. Submit Transaction with multiply signed event

https://medium.com/solana-labs/wormhole-solana-ethereum-bridge-d5502e944acb
https://github.com/certusone/wormhole
https://github.com/certusone/wormhole/blob/main/ethereum/contracts/Wormhole.sol

Attestors submit multiply signed event to source

This is approximately how the Wormhole Solana - Ethereum MainNet bridge works

Advantage:

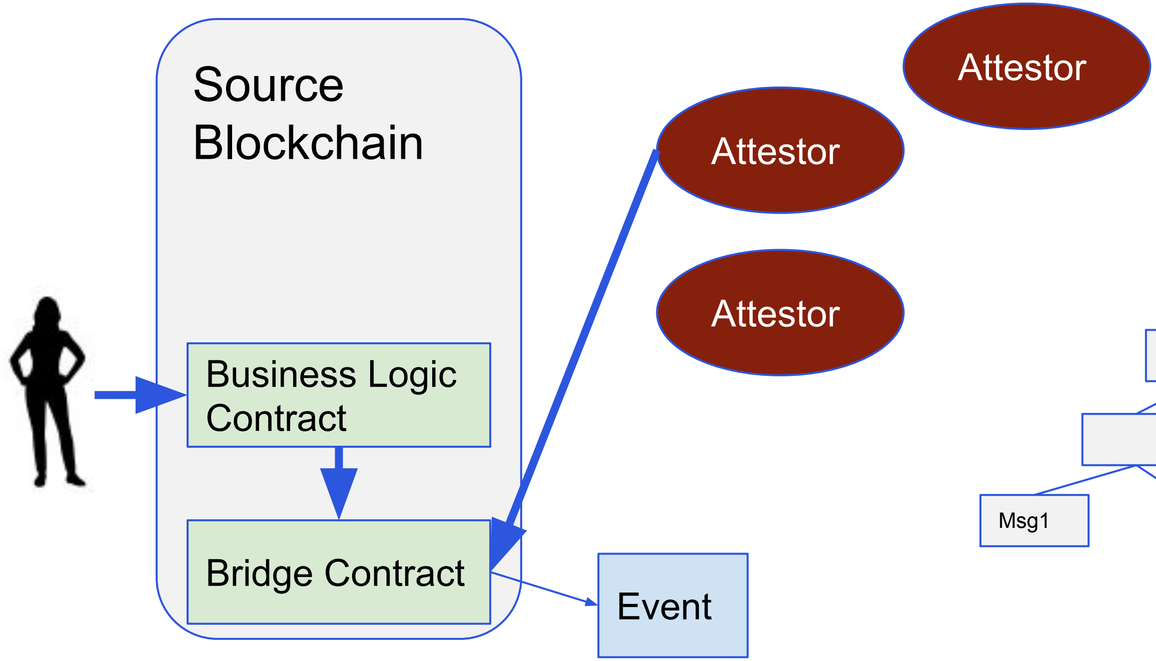
- Relayers do not submit any transactions to the destination blockchain.
- Relayers need to need to understand the target blockchain for an event.
- Users do not interact with Relayers.

Disadvantage:

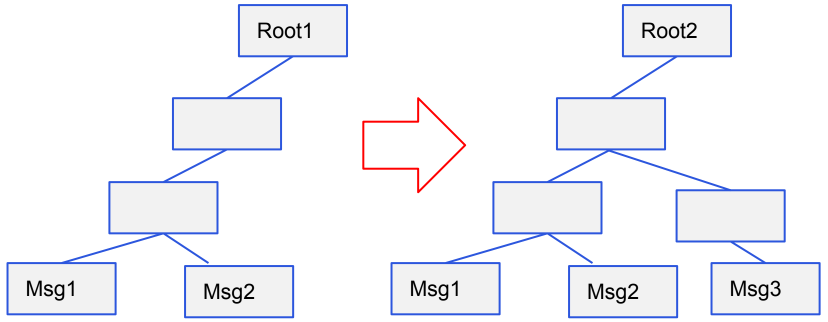
- Relayers need to cooperate.
- User has to be able to submit a transaction on the destination blockchain.

Incremental Merkle Tree

This is approximately how the Celo Optics bridge works



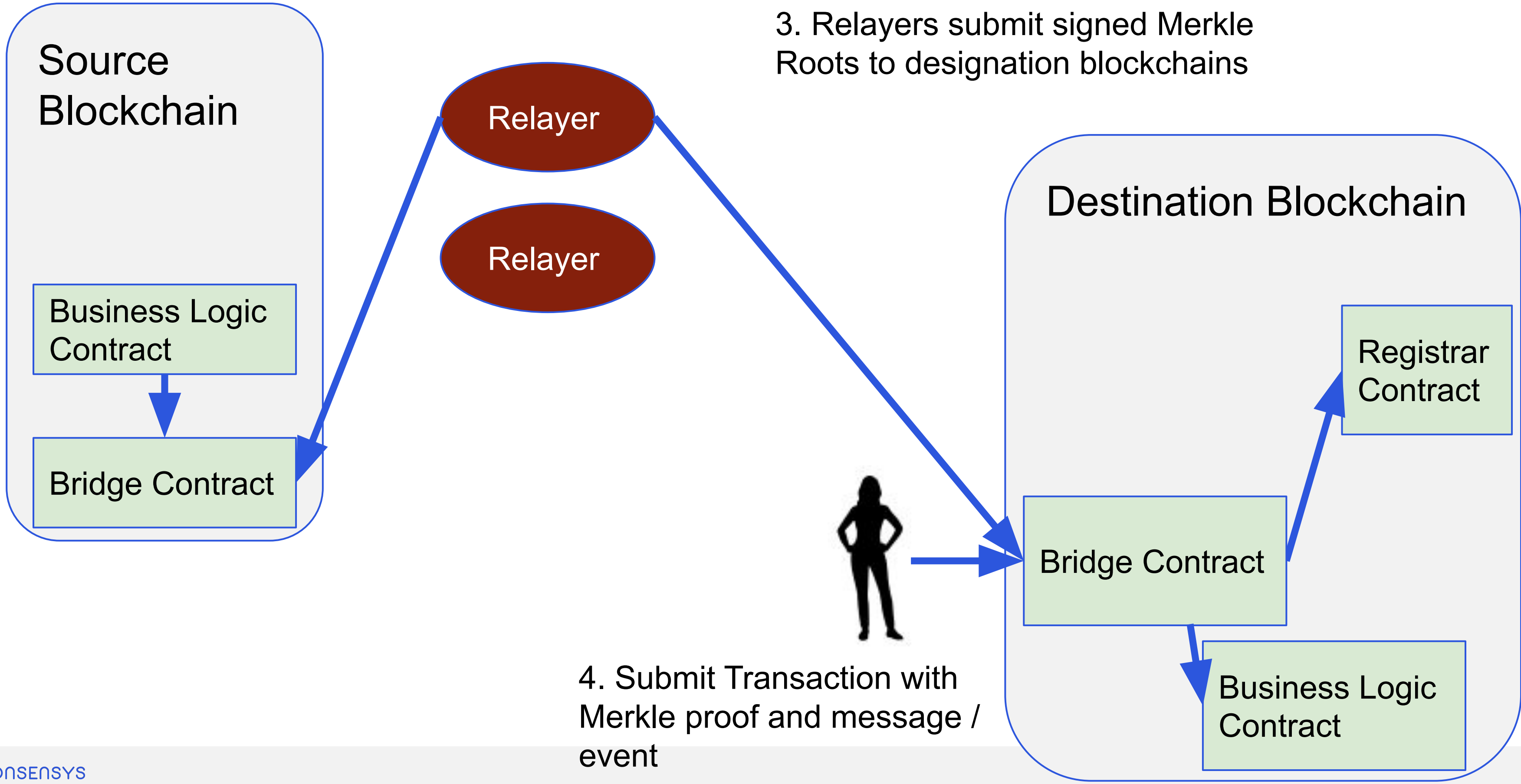
2. Attestors (Updaters) order messages, and submit signed Merkle Roots to source blockchain



1. Submit Transaction that results in a message / event being emitted

Incremental Merkle Tree

This is approximately how the Celo Optics bridge works



3. Relayers submit signed Merkle Roots to designation blockchains

4. Submit Transaction with Merkle proof and message / event

Celo Optics: Optimistic Approach

- Celo Optics relies on only one Relayer transferring a Merkle Root.
  - This is to save gas.
- The Merkle Root can not be used until a “Fraud Window” has expired.
- Observers are expected to submit malicious Merkle Roots back to the source blockchain so that malicious Attestors (Updaters) can be slashed.
- Applications need to check each message and reject / not act on fraudulent messages.
  - This seems like a big imposition on applications.

Attestors submit multiply signed event to source

This is approximately how the Celo Optics bridge works

Advantage:

- Relayers do not submit any transactions to the destination blockchain.
- Relayers need to need to understand the target blockchain for an event.
- Users do not interact with Relayers.

Disadvantage:

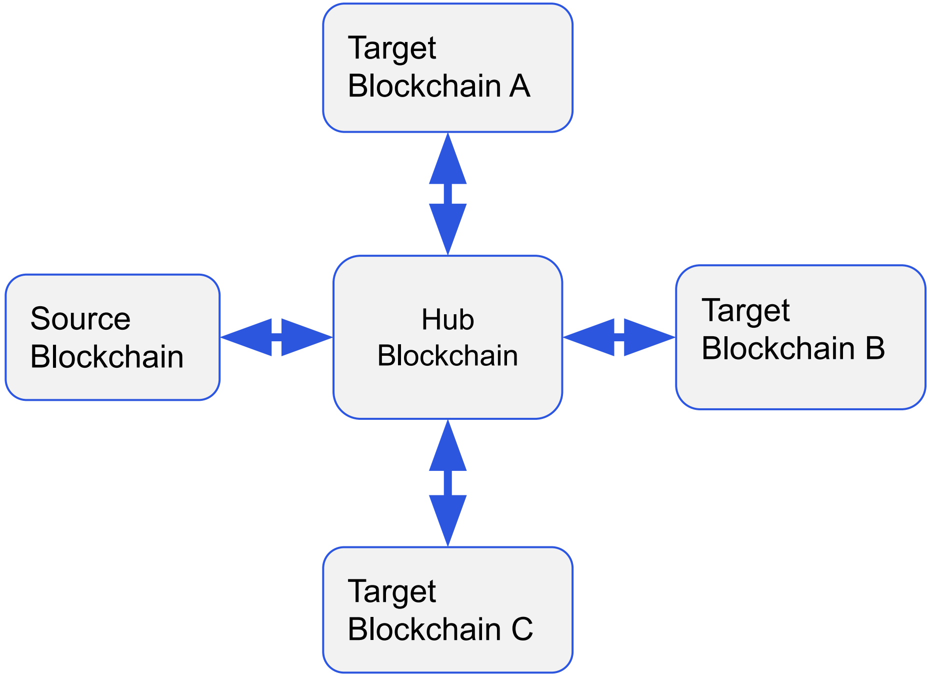
- Relayers need to cooperate.
- User has to be able to submit a transaction on the destination blockchain.

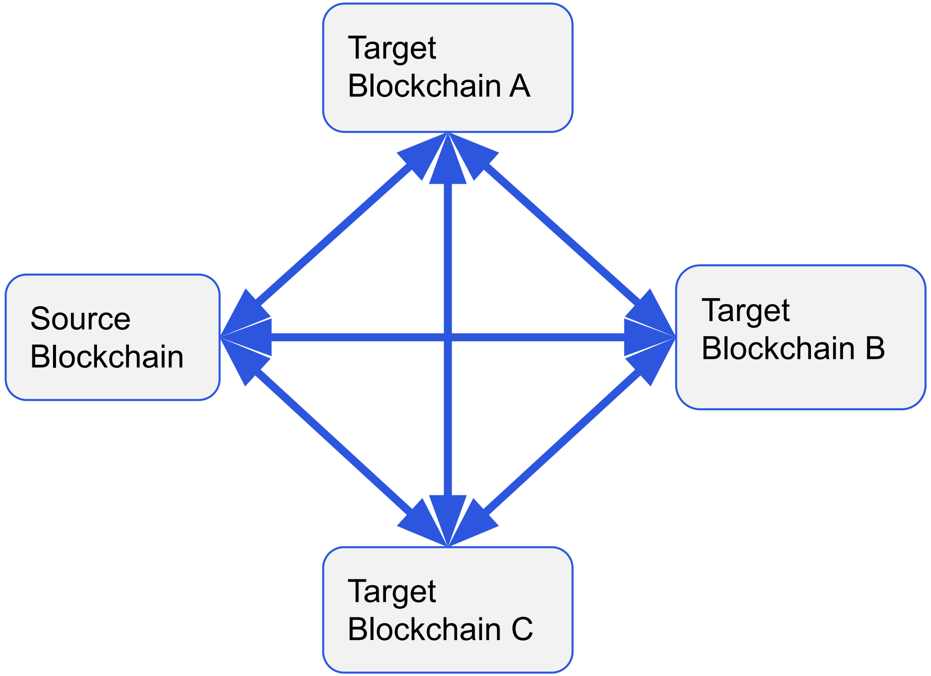
Signature Schemes

- ECDSA: Used for transaction signing in Ethereum. Fast.
- BLS:
  - Signature verification is orders of magnitude slower that ECDSA.
  - Mathematical properties allow signatures and public keys to be added.
- BLS Aggregated Signatures / Threshold Signatures:
  - Attestors independently generate their private keys and public keys.
  - Add multiple signatures, indicate who has signed.
  - Ethereum 2 is using this technique.
- BLS Threshold Signatures:
  - Attestors independently generate private key shares and cooperate to generate a combined public key.
  - Cooperate to create combined signatures.
  - Hides which attestors signed.
  - Complicated set-up.
- Schnorr Aggregated Signatures / Threshold Signatures:
  - Relayers independently generate their private keys and public keys.
  - Combine multiple signatures, indicate who has signed.
  - Wanchain is using this technique.

Hub Blockchains

WanChain, PolyNetwork, and Cosmos use hub blockchain.

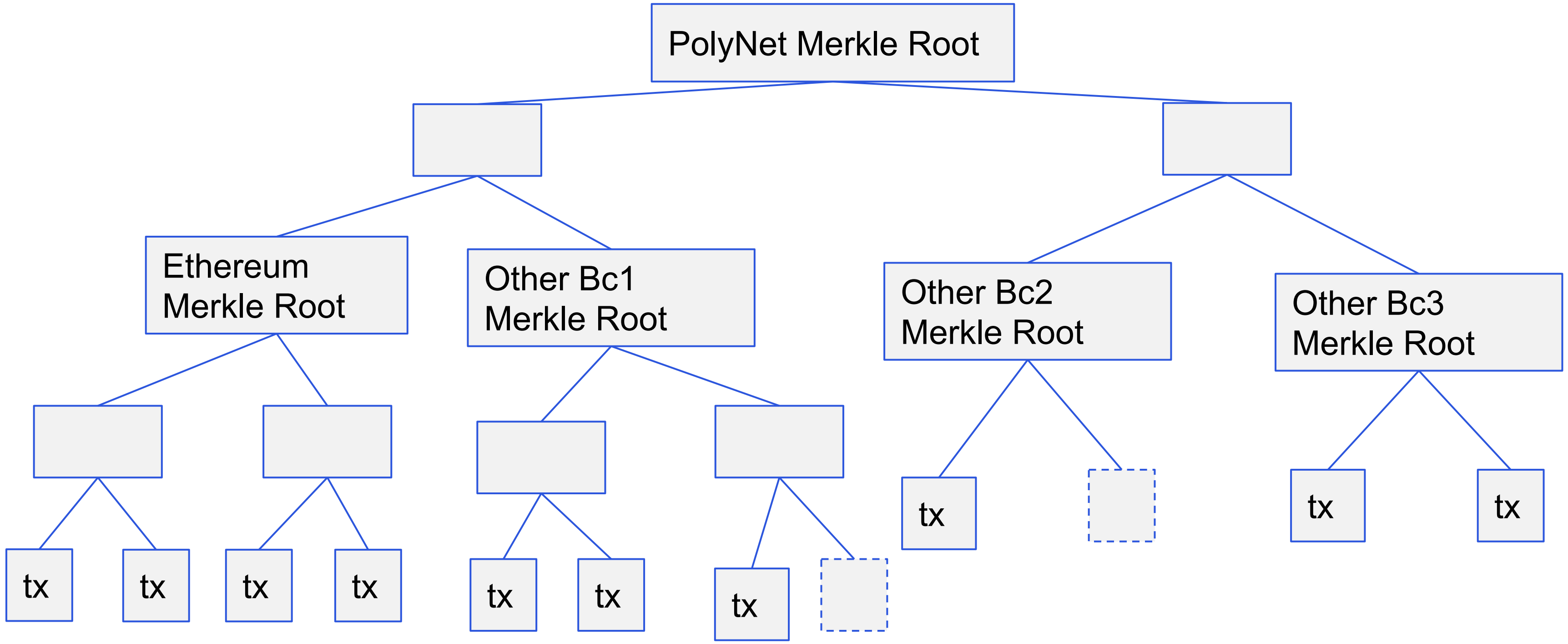




Hub Blockchains

- Reduce the number of bridges required if all blockchains need to be interconnected.
- Do all blockchains need to be connected?
- Hub blockchains will charge $ to use their services.
    - Most bridges, whether via a hub or not are likely to charge $ for their service.
- Hubs increase the latency of crosschain transactions.

Hub Blockchains with Aggregated Merkle Tree



Staking and Slashing

Attestors / Relayers can be required to stake some $.

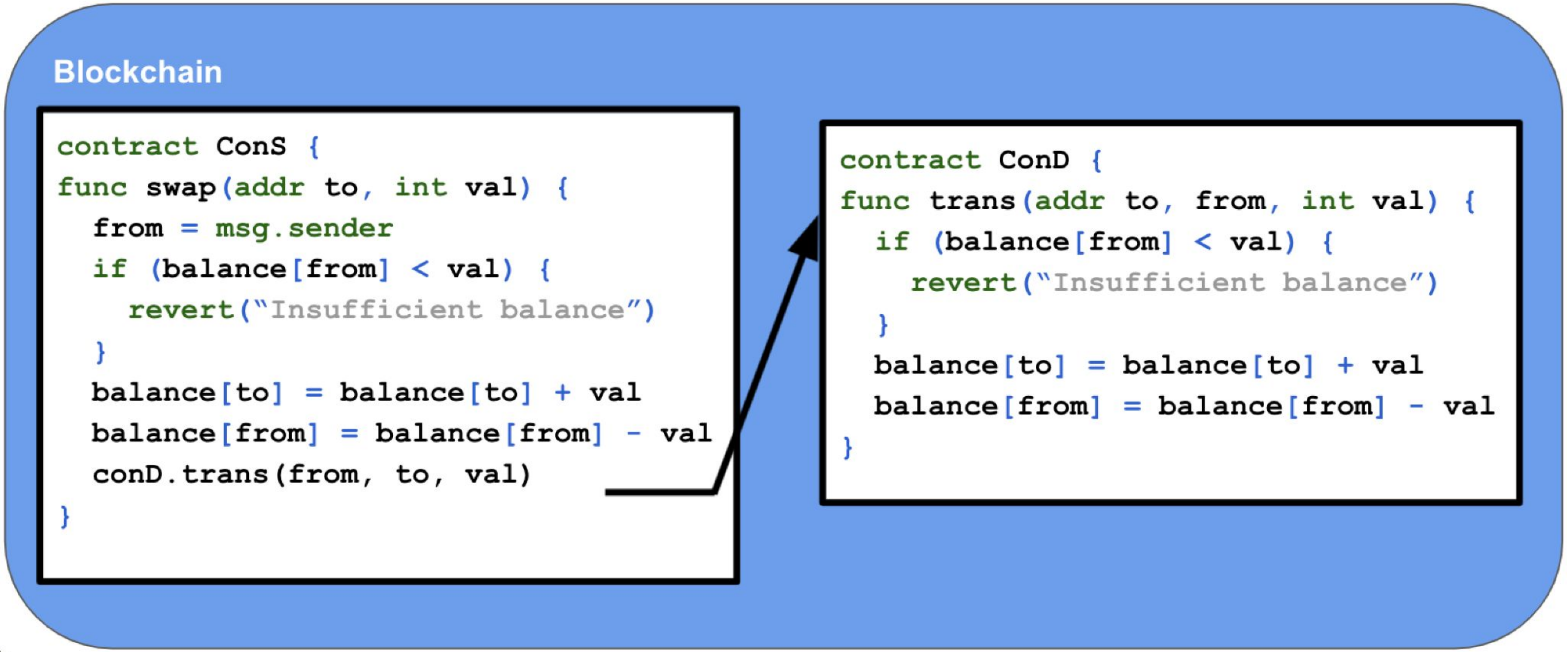
Questions to consider:

- How much is being staked?
- How does the amount being staked relate to the amount that could be stolen?
- In what situations can a relayer be slashed?
- How is the misbehaviour proven? That is, what cryptographic enforcement is there?

CONSENSYS

Design Choices: Crosschain Function Call Layer

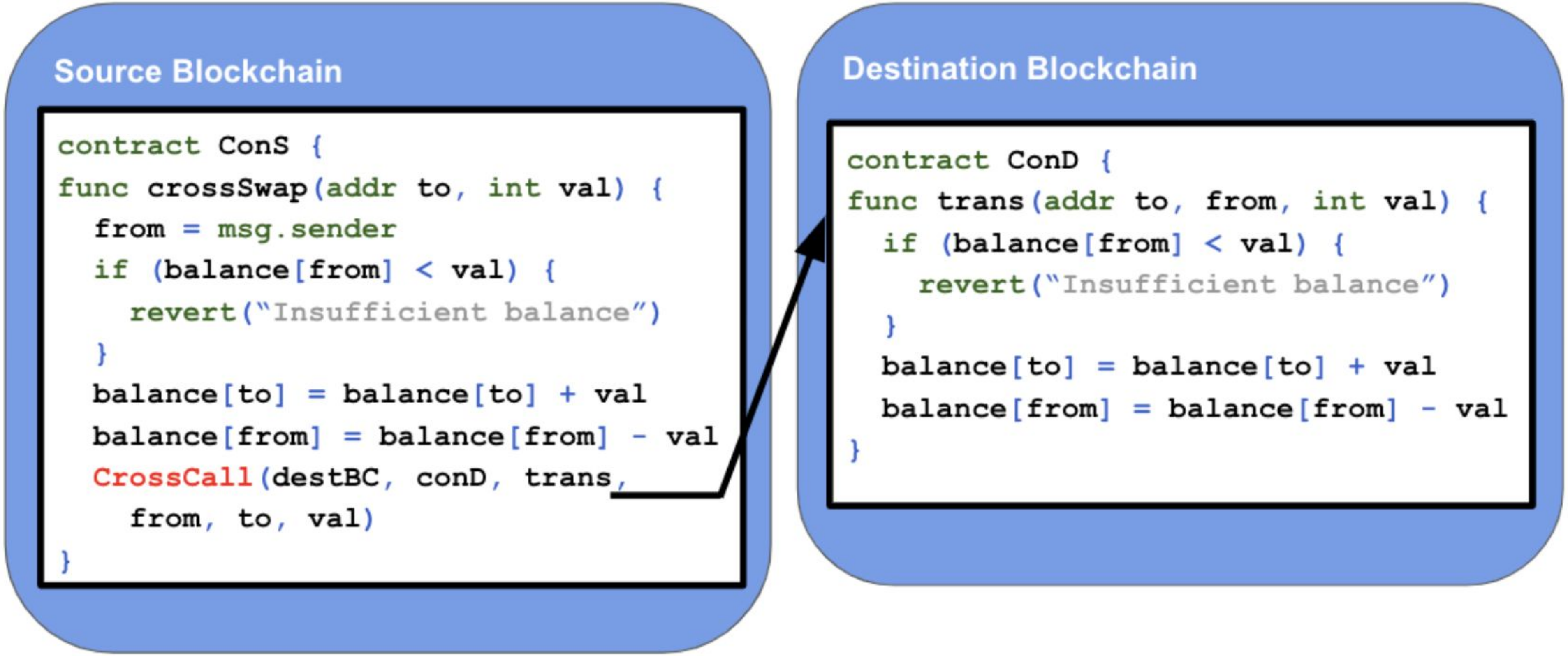
Function Calls: Single Blockchain



```
contract ConS {
func swap(addr to, int val) {
  from = msg.sender
  if (balance[from] < val) {
    revert("Insufficient balance")
  }
  balance[to] = balance[to] + val
  balance[from] = balance[from] - val
  conD.trans(from, to, val)
}
```

```
contract ConD {
func trans(addr to, from, int val) {
  if (balance[from] < val) {
    revert("Insufficient balance")
  }
  balance[to] = balance[to] + val
  balance[from] = balance[from] - val
}
```

Function Calls: Crosschain



```
contract ConS {
func crossSwap(addr to, int val) {
  from = msg.sender
  if (balance[from] < val) {
    revert("Insufficient balance")
  }
  balance[to] = balance[to] + val
  balance[from] = balance[from] - val
  CrossCall(destBC, conD, trans,
    from, to, val)
}
```

```
contract ConD {
func trans(addr to, from, int val) {
  if (balance[from] < val) {
    revert("Insufficient balance")
  }
  balance[to] = balance[to] + val
  balance[from] = balance[from] - val
}
```

Design Considerations

- Atomic (GPACT) OR non-atomic (SFC)
- Bridge contract linkage between chains.
- Replay protection.
- Timeout / age events / transactions.
- Application authentication and access control.
- Arbitrary execution OR known functions.
- Pausability.

Function Call Layer Design: Atomic or Not atomic

Reasons why a transaction might fail:

- The **revert** in the trans function could be triggered.
- The account that submitted the transaction on the destination account might not have enough Ether to pay for the gas to execute the transaction.
- The account might not have permission to execute the transaction. This could be at the Solidity contract level, where permissioning may have been configured to limit which accounts can call a function. In a permissioned blockchain network, only certain accounts might be permissioned to submit transactions.
- The transaction may have been submitted with a gas price that is too low given the current gas price required to have a transaction included in a block.
- The transaction may be incorrectly configured, for instance with an incorrect nonce value.

Function Call Layer Design: GPACT vs Non-Atomic

GPACT	Non-Atomic
Atomic updates across blockchains.	----
Crosschain transaction fails handled by protocol.	Crosschain transaction failures need to be handled within the application.
Higher gas costs (10x single blockchain)	Lower gas costs (3x single blockchain)
Contracts designed with locking in mind	Any contract
Composable programming model that application developers are accustomed to. From the perspective of the application developer, ● All transactions occur simultaneously. ● Functions return values immediately.	Non-standard programming model: ● Each segment of the overall crosschain transaction is separate. ● Functions can not return values.

Function Call Layer Design: GPACT vs Non-Atomic

These numbers are implementation specific. Fewer checks / less functionality means closer to single blockchain performance.

GPACT	Non-Atomi
Atomic updates across blockchains.	----
Crosschain transaction fails handled by protocol.	Cross[illegible]ain t[illegible] ha[illegible]ed within the application.
Higher gas costs (10x single blockchain)	Lower gas costs (3x single blockchain)
Contracts designed with locking in mind	Any contract
Composable programming model that application developers are accustomed to. From the perspective of the application developer, ● All transactions occur simultaneously. ● Functions return values immediately.	Non-standard programming model: ● Each segment of the overall crosschain transaction is separate. ● Functions can not return values.

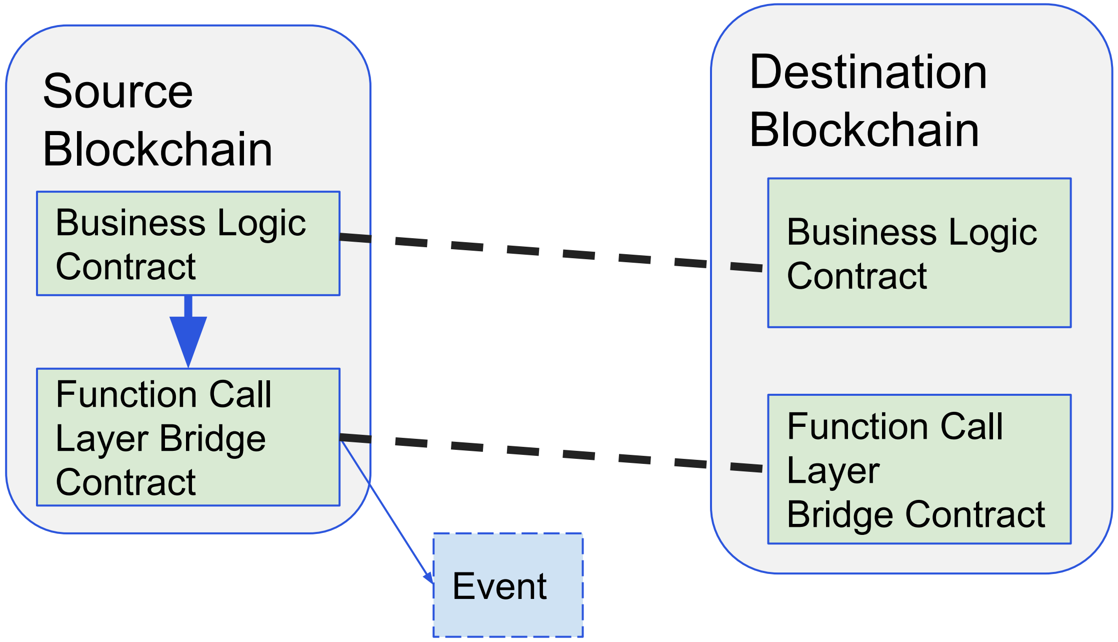
CONSENSYS

Design Choices: Crosschain Application Layer

Crosschain Application Design: Access Control

Access control:

- Use access control provided by function call layer to implement an **Allow List** of application contracts on other blockchains that are authorised to call the function.



Crosschain Application Layer Design: Pausable

- All functions on the application bridge should be able to be stopped.
- The functions to pause and unpause the bridge need to be access control protected.

- If the Function Call Bridge is used by multiple applications, you can not rely on the Function Call Layer “Pausable” feature to pause the application bridge.

For Applications using non-Atomic Function Calls

Admin control to allow failed transactions to be rolled back:

```
function adminTransfer(
  address _erc20Contract,
  address _recipient,
  uint256 _amount) onlyReallyTrustedAdmin external {
  ...
}
```

1. Who is this trusted admin?
2. What if the $ under control is huge?
3. Use a multi-sig wallet!

For GPACT: Locking Design

If a storage location in an application contract needs to be locked...

Lock whole contract	**Lock just the storage location**
Less gas	More gas
Less complex	More complex (though template contracts handle the complexity)
One crosschain transaction at a time	Many crosschain transactions at a time

Lock on read?

Isolation: return same read value within a transaction.

ERC 20 Mass Conservation vs Minting and Burning

- ERC 20 Minting Burning:
  - The Function Call Bridge contract is likely to have the Minter role.
  - If there is a crosschain bug, an attacker could mint coins.
- ERC 20 Mass Conservation:
  - The Function Call Bridge contract could be given control over some number of tokens ("active tokens").
  - Move ownership of tokens from the bridge to another account when is holds too many tokens.
  - Give tokens to the bridge when it is running out of token.
  - If there is a crosschain bug, the worst an attacker could steal is the "active tokens".

Limiting value on a bridge

- How much should validators stake? US$50K? How many validators?
  - Stake x signing threshold = amount that can be slashed.
- The amount of value on the bridge should be less than the amount that is ~~staked~~ that can be slashed.

Questions:

- Which bridge / which layer?
  - Messaging layer?
  - Function call layer?
  - Application layer?
- How could a Messaging layer or Function Call layer implementation know about the value on the bridge?
- How does this work with ERC 721 NFT tokens?
- What if the bridge is a Trade - Finance application?
- How are transactions on the bridge temporarily rejected, held, or paused?

CONSENSYS

Summary

Summary

Having a layered architecture with **clearly defined interfaces between layers** has the following advantages:

- **Interoperability**: Applications will work with a variety of Function Call implementations that will work with a variety of Messaging implementations.
- **Flexibility**: different Crosschain Message Verification technique could be used for each blockchain / roll-up used in an overall crosschain function call.
- **Infrastructure Reuse**: Different Crosschain Function Call techniques can share the same deployed Crosschain Message Verification infrastructure.
- **Focus on what you are good at**: Organisations can focus on creating solutions for a single part of the protocol stack stack. That is, rather than having to create the entire stack, a company might choose to focus on an application, a better Function Call approach, or some Messaging infrastructure.
- **Experimentation**: Not having to build the entire protocol stack should make experimentation much easier.