Replay Attacks and Defenses Against Cross-shard Consensus in Sharded Distributed Ledgers

Alberto Sonnino
A set of nodes
Byzantine Fault Tolerance

$\frac{2}{3} > \frac{2}{3}$
Consensus
State Sharding
State Sharding
An example transaction

\[ T(x_1, x_2) \rightarrow (y_1, y_2, y_3) \]
State Sharding

An example transaction

\[ T(x_1, x_2) \rightarrow (y_1, y_2, y_3) \]
State Sharding
Only two acceptable final states
Cross-Shard Consensus
How do shards communicate with each other?
Chainspace: A Sharded Smart Contracts Platform

S-BAC

Atomix

OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding

Abstract

Chainspace is a novel sharded smart contract platform that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability. This paper makes the following contributions:

1. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

2. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

3. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

4. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

5. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

6. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

7. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

8. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

9. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

10. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

11. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

12. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

13. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

14. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

15. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

16. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

17. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

18. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

19. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

20. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

21. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

22. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

23. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

24. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

25. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

26. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

27. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

28. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

29. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

30. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

31. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

32. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

33. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

34. A novel sharded smart contract platform called Chainspace, that allows for horizontal scalability, control of which part of the infrastructure needs to be trusted on a per-contract basis, and also allows for horizontal scalability.

The scalability of distributed ledgers (DLs), in both total throughput and number of participants, is by now a well-known security and decentralization challenge. Many approaches have been proposed for sharded DLs, however, they all suffer from permissionless decentralization. To achieve this goal, OmniLedger introduces Atomix, a two-phase client-driven "lock/unlock" protocol that ensures that clients can either fully commit a transaction or unlock state affected by partially completed transactions.

Verifiable and efficient: Atomix allows for high-integrity and high-privacy applications.

Further, it can handle varying transaction loads.

The proven and obvious approach to building "permissionless" blockchains is the Nakamoto consensus [13], [14]. These approaches rely on a "proof-of-work" (PoW) consensus algorithm to enforce that each block in the blockchain is verified by "mining" the block, where a "proof of work" is a random number that must be included in the block [15]. This approach works well for public (i.e., permissionless) ledgers, since it ensures that the ledger is secure, tamper-proof, and cannot be changed. However, the security model of Chainspace is different; it focuses on privacy-friendly smart contracts and evaluates their performance.

S-BAC and Atomix have been implemented as proof-of-concept systems. S-BAC has been implemented in Standard ML, while Atomix has been implemented in Go. Both systems are publicly available on GitHub [16]. We conclude with a discussion of related work, and Section X explores future directions for research in this area.

We introduce OmniLedger, the first DL architecture that presents an overview of Chainspace; Section II presents an overview of Chainspace; Section III introduces the novel distributed atomic commit protocol that ensures that clients can either fully commit a transaction or unlock state affected by partially completed transactions.

S-BAC and Atomix have been implemented as proof-of-concept systems. S-BAC has been implemented in Standard ML, while Atomix has been implemented in Go. Both systems are publicly available on GitHub [16]. We conclude with a discussion of related work, and Section X explores future directions for research in this area.

We introduce OmniLedger, the first DL architecture that presents an overview of Chainspace; Section II presents an overview of Chainspace; Section III introduces the novel distributed atomic commit protocol that ensures that clients can either fully commit a transaction or unlock state affected by partially completed transactions.

S-BAC and Atomix have been implemented as proof-of-concept systems. S-BAC has been implemented in Standard ML, while Atomix has been implemented in Go. Both systems are publicly available on GitHub [16]. We conclude with a discussion of related work, and Section X explores future directions for research in this area.
S-BAC

$T(x_1, x_2) \rightarrow (y_1, y_2, y_3)$

client

shard 1

shard 2

shard 3
S-BAC

\[ T(x_1, x_2) \rightarrow (y_1, y_2, y_3) \]
S-BAC

\[ T(x_1, x_2) \rightarrow (y_1, y_2, y_3) \]

client → shard 1 → shard 2 → shard 3
$T(x_1, x_2) \rightarrow (y_1, y_2, y_3)$

- **S-BAC**
- Client
- Shard 1
- Shard 2
- Shard 3
  - Lock $X_1, X_2$
S-BAC

\[ T(x_1, x_2) \rightarrow (y_1, y_2, y_3) \]
S-BAC

\[ T(x_1, x_2) \rightarrow (y_1, y_2, y_3) \]

delete $X_1, X_2$; create $Y_1, Y_2$
S-BAC

\[ T(x_1, x_2) \rightarrow (y_1, y_2, y_3) \]
atomix

$T(x_1, x_2) \rightarrow (y_1, y_2, y_3)$

client

shard 1

shard 2

shard 3
Insecure under parallel composition
Attacks

**Double spend any object**

- Does not need to collude with any node
- Acts as client or passive observer
- Re-orders network messages (not always needed)
Attack against S-BAC

Double-spend $X_1$

\[ T(x_1, x_2) \rightarrow (y_1, y_2, y_3) \]
Attack against S-BAC

Double-spend $X_1$

$$T'(\overline{x_1, x_2}) \rightarrow (y_1, y_2, y_3)$$
Attack against S-BAC

Double-spend $X_1$

$T'(\widetilde{x}_1, x_2) \rightarrow (y_1, y_2, y_3)$

s1 \rightarrow BFT

s2 \rightarrow BFT

s3
Attack against S-BAC

Double-spend $X_1$

$T'((\overline{x_1}, x_2) \rightarrow (y_1, y_2, y_3)$

$c \leftarrow \text{BFT}$

$s_1 \leftarrow \text{BFT}$

$s_2 \leftarrow \text{BFT}$

$s_3 \leftarrow \text{lock } X_2$
Attack against S-BAC

Double-spend $X_1$

\[ T'(\overline{x_1}, x_2) \rightarrow (y_1, y_2, y_3) \]

- c
- s1
- s2
- s3

pre-abort($T'$)
pre-accept($T'$)

lock $X_2$
Attack against S-BAC

Double-spend $X_1$

$T'(\tilde{x}_1, x_2) \rightarrow (y_1, y_2, y_3)$

$c$  

$s_1$  

$s_2$  

$s_3$  

$T(x_1, x_2) \rightarrow (y_1, y_2, y_3)$

$c$  

$s_1$  

$s_2$  

$s_3$  

abort($T$)

pre-accept($T$)

pre-abort($T'$)

lock $X_2$
Attack against S-BAC

Double-spend $X_1$

$T'(\bar{x}_1, x_2) \rightarrow (y_1, y_2, y_3)$

$T(x_1, x_2) \rightarrow (y_1, y_2, y_3)$

lock $X_2$

pre-accept($T'$) from shard 1

pre-accept($T$)

pre-abort($T'$)

pre-abort($T$)

abort($T$)
Attack against S-BAC

Double-spend $X_1$

$T'(\overline{x_1}, x_2) \rightarrow (y_1, y_2, y_3)$

$T(x_1, x_2) \rightarrow (y_1, y_2, y_3)$

$c$  

$s_1$  

$s_2$  

$s_3$  

lock $X_2$

pre-accept($T'$)  

pre-accept($T'$)  

pre-accept($T'$)  

unlock $X_2$

abort($T'$)  

abort($T$)  

abort($T'$)  

Pre-accept($T$)

from shard 1
Attack against S-BAC

Double-spend $X_1$

$T^*(x_1) \rightarrow (y_*)$

client

shard 1

BFT

10
Attack against S-BAC
Double-spend $X_1$

client

$T(x_1, x_2) \rightarrow (y_1, y_2, y_3)$

shard 1

$\text{BFT} \rightarrow \text{BFT}$

shard 2

$\text{BFT} \rightarrow \text{BFT}$

attacker

pre-accept($T$)

pre-abort($T$)
Attack against S-BAC

Double-spend $X_1$

$T(x_1, x_2) \rightarrow (y_1, y_2, y_3)$

client

shard 1

shard 2

shard 3

attacker
Attack against S-BAC

Double-spend $X_1$

$T(x_1, x_2) \rightarrow (y_1, y_2, y_3)$
Attack against S-BAC
Double-spend $X_1$

<table>
<thead>
<tr>
<th></th>
<th>Before attack</th>
<th>After attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$X_2$</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>$Y_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Y_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Y_3$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Byzcuit
S-BAC + Atomix