IC3 The Initiative For CryptoCurrencies & Contracts

KING'S College LONDON

You sank my battleship!

A case study to evaluate state channels as a scaling solution for cryptocurrencies.

Patrick McCorry

with lots of help from Andrew Miller, Iddo Bentov, Surya Bakshi, Sarah Meiklejohn, Ranjit Kumaresan, Christopher Cordi, Chris Buckland, Karl Wust.

paddyucl

Cryptocurrencies do not scale.

7 tps (sort of)

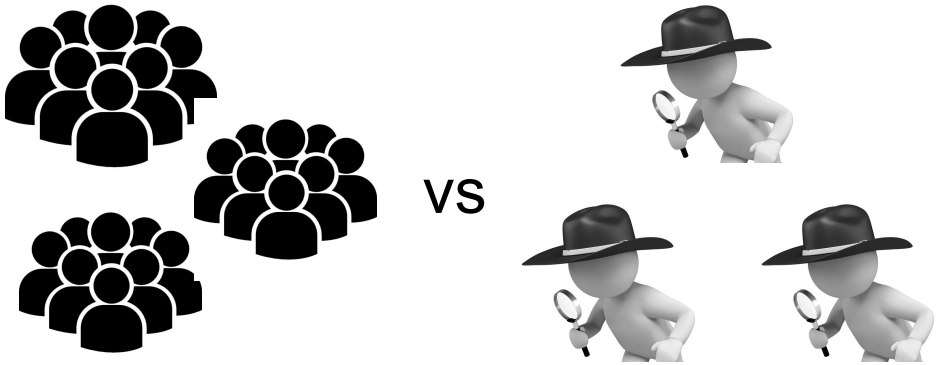
1 MB blocks

12 tps (sort of)

Complexity of transaction

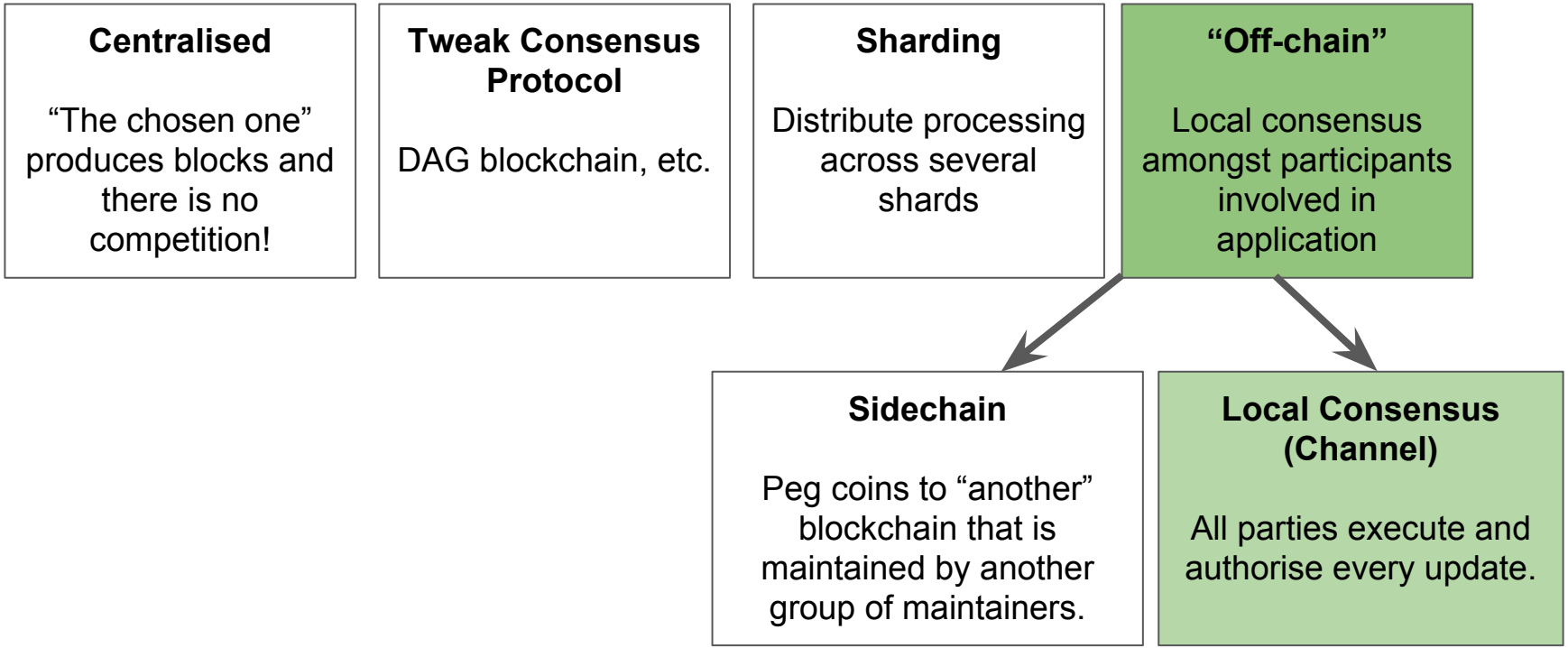
https://github.com/stonecoldpat/anonymousvoting

What is the problem?

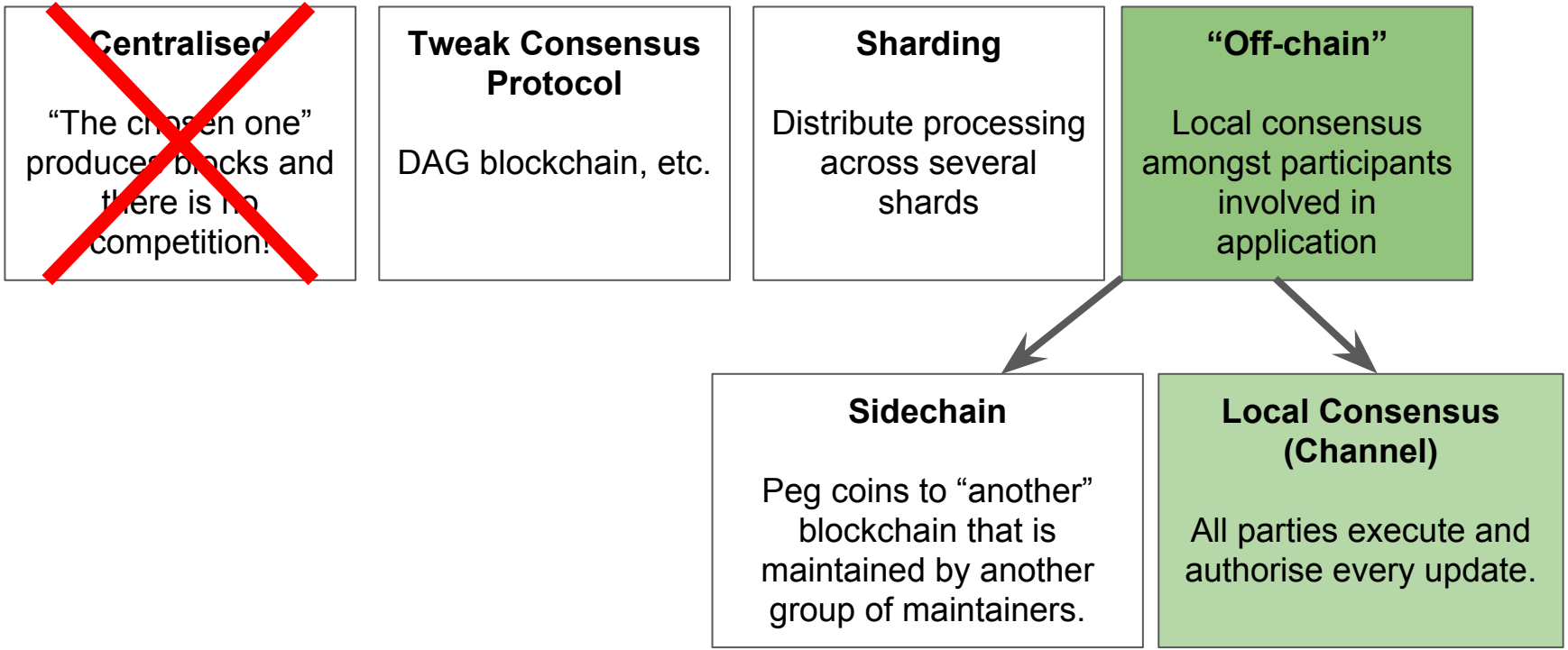


“Decentralisation and Public Verifiability”
Large user-base vs Large set of validators

OK. How can we scale?



OK. How can we scale?



What is a “channel”?

Every party **collectively authorises a new state** of an application **locally amongst themselves.**

The blockchain acts as a root of trust to **guarantee safety**

Essentially, each party is refunded the coins they deserve if one party does not co-operate off-chain.

Payment channels

Only re-distribute deposit

Spilman Payment Channels	Duplex Micropayment Channels	Lightning Channels	Raiden Channels

Payment channels

Only re-distribute deposit

Spilman Payment Channels	Duplex Micropayment Channels	Lightning Channels	Raiden Channels

State channels

gaming, voting, auctions, etc

Sprites Channels	Perun Channels	Counterfactual	Under Construction

What modifications are required **before an application can be deployed as a state channel**?

What **applications make sense** in a state channels?

What are the **inherent weaknesses**?

Battleship is two player and turn based game

Set Up

Boths players place ships on their boards

Game Play

One player attacks a cell (i.e. A,0)

Counterparty opens cell as water or ship location.

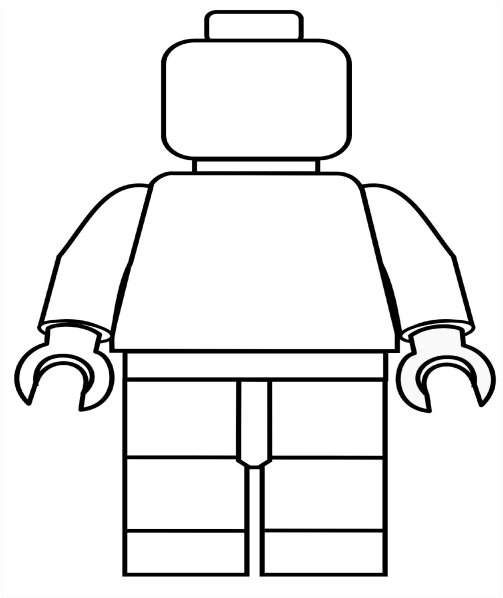
If attacker is hits final ship piece - the counterparty should also declare their ship as sunk!

End Game

After one player has sunk ships, both players reveal their board.

A Trusted Third Party with Public State

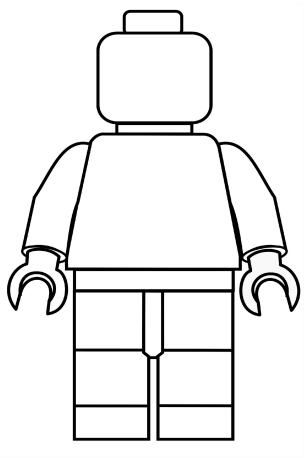
Hey, I'm a smart contract.



I promise you the following:

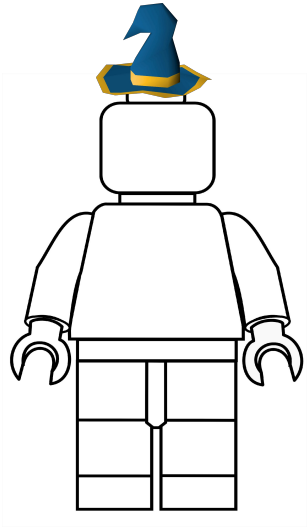
1. I will never modify or change your code.
2. I will always run the code you tell me too (assuming the code itself allows me!).
3. I will never let code execution "stop half way" it is ALL or NOTHING with me.
4. I like to gossip and I can't keep secrets - Everything you tell me will be public knowledge.

Two Smart Contracts



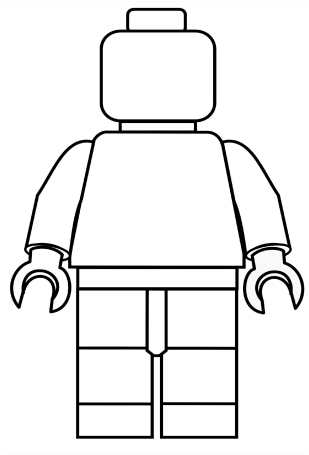
Application (“Battleship”) Contract

Lets players play the battleship game via the blockchain



State Channel Contract

Decides the “final game state” after players have finished playing amongst themselves



Application (“Battleship”) Contract

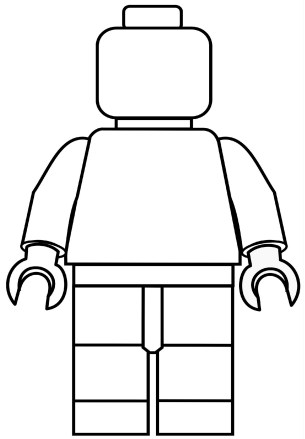
Function lock()

When approved by all parties, lock alll functionality within the app and launch the state channel

Function unlock(state, r)

After the state channel is closed, accepts “full state” and re-enables functionality

A new life-cycle

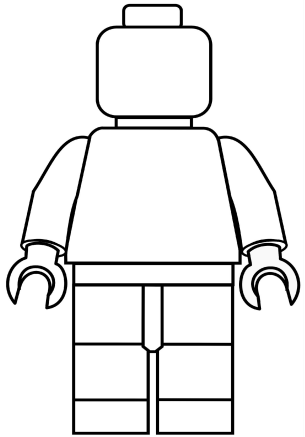


Application Contract

A new life-cycle

1. **Everyone** signs the “lock” message

σ_A, σ_B, **“lock”**



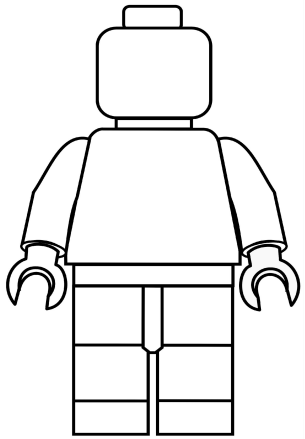
Application Contract

A new life-cycle

1. **Everyone** signs the “lock” message

2. **Application contract** receives “lock” message.

σ_A, σ_B, **“lock”**



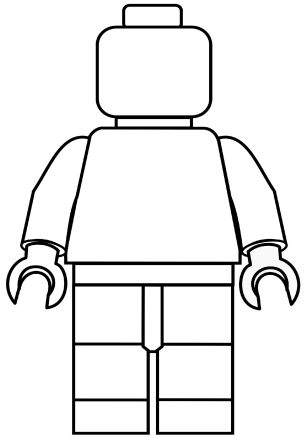
Application Contract

A new life-cycle

1. **Everyone** signs the “lock” message

2. **Application contract** receives “lock” message.

σ_A, σ_B, **“lock”**



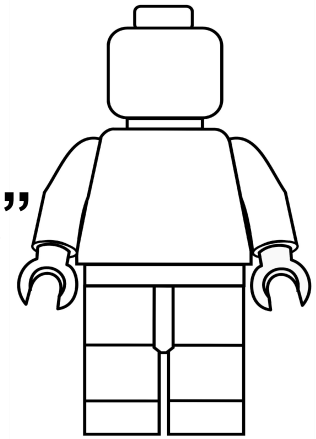
Application Contract

A new life-cycle

1. **Everyone** signs the “lock” message

2. **Application contract** receives “lock” message.

σ_A, σ_B, **“lock”**

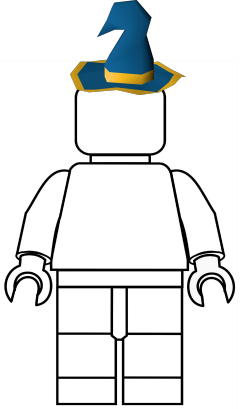


Application Contract

A new life-cycle

1. **Everyone** signs the “lock” message

2. **Application contract** receives “lock” message.
   - Locks ALL functionality
   - Instantiates state channel contract



State Channel Contract

Application Contract

Application is locked on the blockchain

How do we progress an application off-chain?

State of game

Alice’s turn to take a shot at bob’s ship

State 0

Turn in Game: Alice
Counter: 0

If you squint hard enough…

Alice is shooting from a real battleship

State 0

Turn in Game: Alice
Counter: 0


61

State Transition

Alice's shot is "locked in"

State 0	State 1
Turn in Game: Alice **Counter:** 0	**Turn in Game:** Bob **Counter:** 1

Both parties must agree the game's new state

Alice shot is locked in and Bob must open the cell on his board.

State 0	State 1
Turn in Game: Alice **Counter:** 0	**Turn in Game:** Bob **Counter:** 1

Authorising New State

Both parties sign new state and exchange signatures

State 0	State 1
Turn in Game: Alice **Counter:** 0	**Turn in Game:** Bob **Counter:** 1

Authorising New State

Both parties sign new state and exchange signatures

State 0	State 1
Turn in Game: Alice **Counter:** 0	**Turn in Game:** Bob **Counter:** 1

State 1 is AUTHORISED

Each party has a signature from every other party

State 0	State 1
Turn in Game: Alice **Counter:** 0	**Turn in Game:** Bob **Counter:** 1

Bob’s turn!

Bob’s must open the attacked cell

State 0	State 1
Turn in Game: Alice **Counter:** 0	**Turn in Game:** Bob **Counter:** 1

Has Alice hit a ship….?


P121

Command Outcome

Alice hit Bob's battleship
and the final piece on his board!

State 0	State 1
Turn in Game: Alice **Counter:** 0	**Turn in Game:** Bob **Counter:** 1

Both parties must agree the game's new state

Alice has won the game!

State 0	State 1	State 2
Turn in Game: Alice **Counter:** 0	**Turn in Game:** Bob **Counter:** 1	**Winner:** Alice **Counter:** 2

Authorising New State

Both parties sign new state and exchange signatures

State 0	State 1	State 2
Turn in Game: Alice **Counter:** 0	**Turn in Game:** Bob **Counter:** 1	**Winner:** Alice **Counter:** 2

Authorising New State

Both parties sign new state and exchange signatures

State 0	State 1	State 2
Turn in Game: Alice **Counter:** 0	**Turn in Game:** Bob **Counter:** 1	**Winner:** Alice **Counter:** 2

State 2 is AUTHORISED

Each party has a signature from every other party

State 0	State 1	State 2
Turn in Game: Alice **Counter:** 0	**Turn in Game:** Bob **Counter:** 1	**Winner:** Alice **Counter:** 2

Not mentioned:

All parties sign the “hash” of every new state.

Not important for learning about how state channels work,
But allows us to build an “independent” state channel contract!

What if one party does not cooperate and does not sign the new state update?

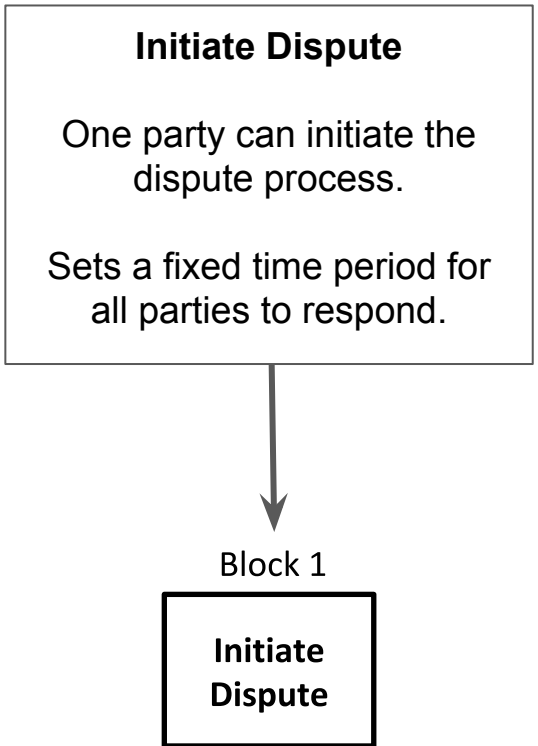
Time to self-enforce it via the dispute process!

The Dispute Process

What if everyone doesn’t authorise the new state?

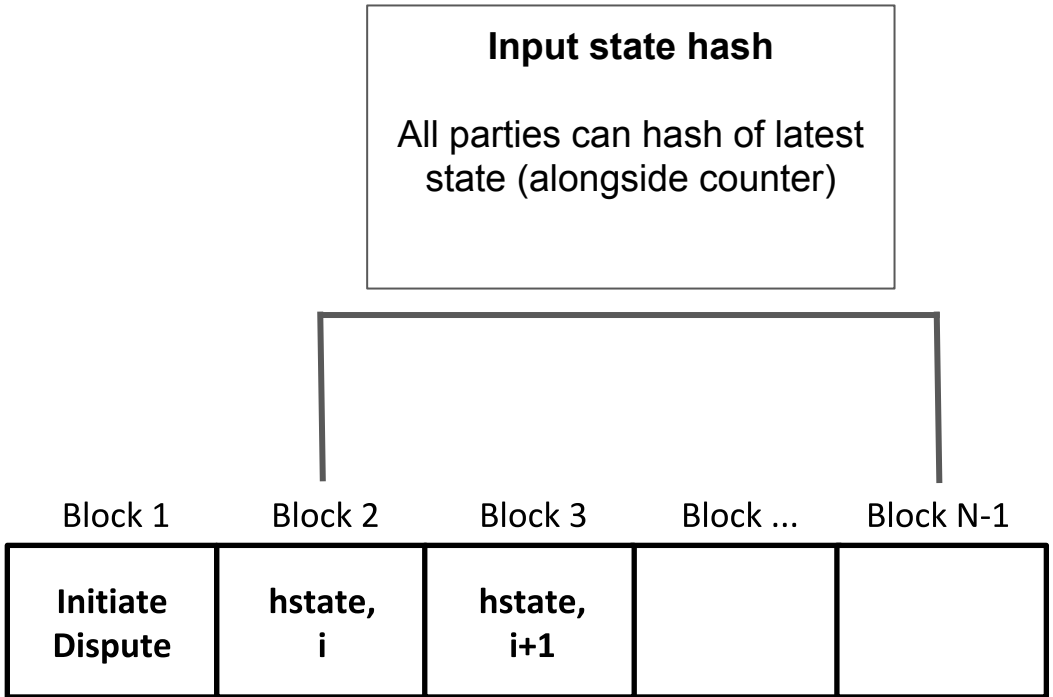
The Dispute Process

What if everyone doesn't authorise the new state?



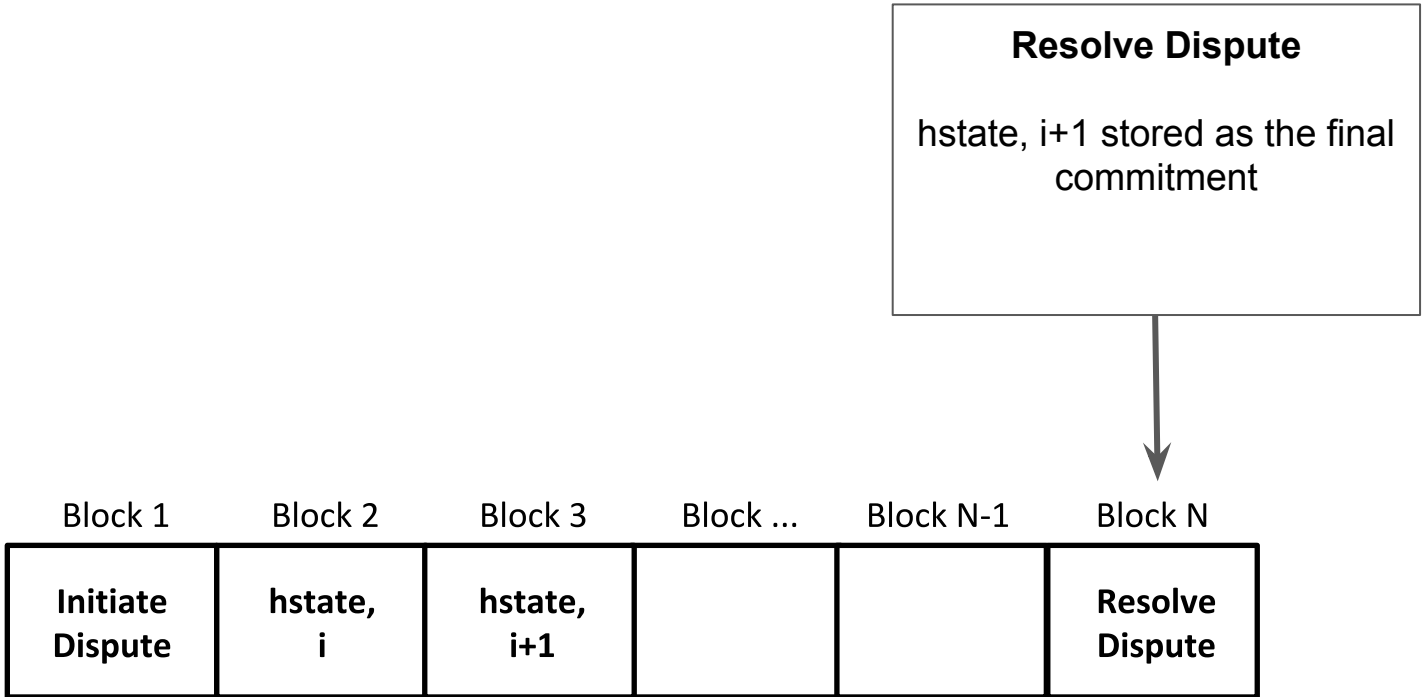
The Dispute Process

What if everyone doesn't authorise the new state?



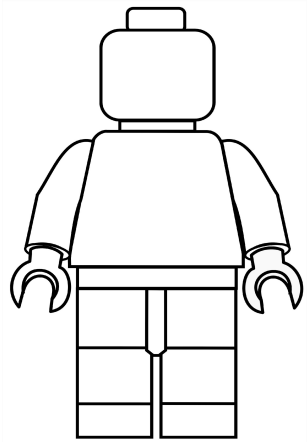
The Dispute Process

What if everyone doesn't authorise the new state?

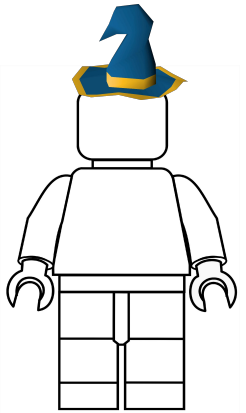


Re-activation!

unlock(state, r)



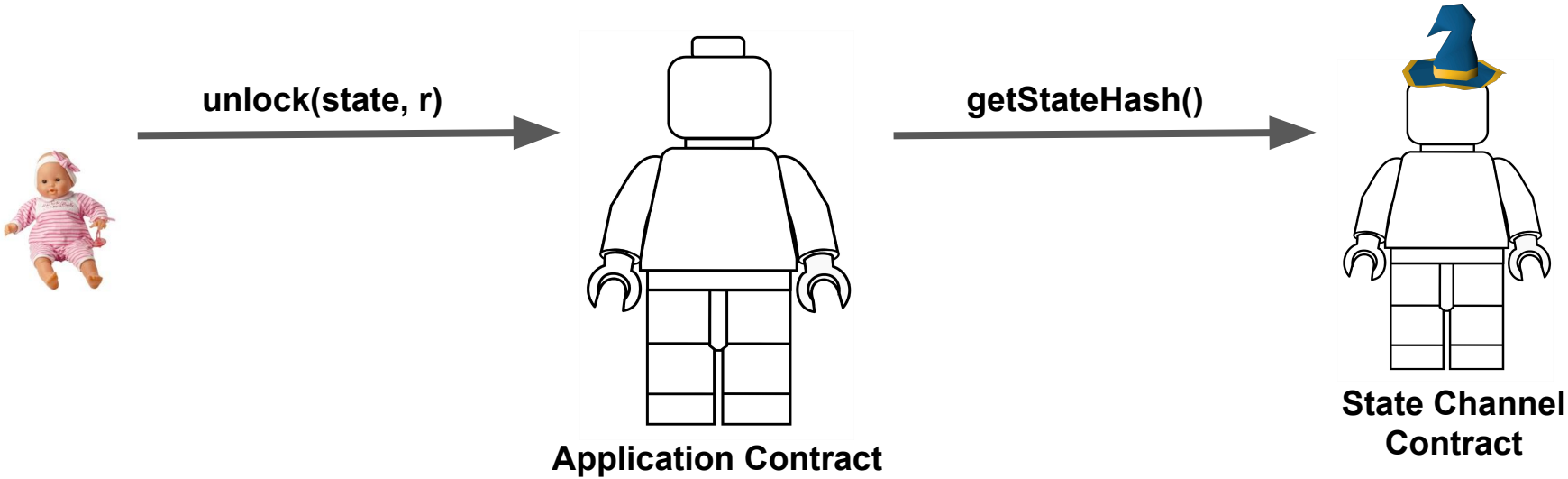
Application Contract



State Channel Contract

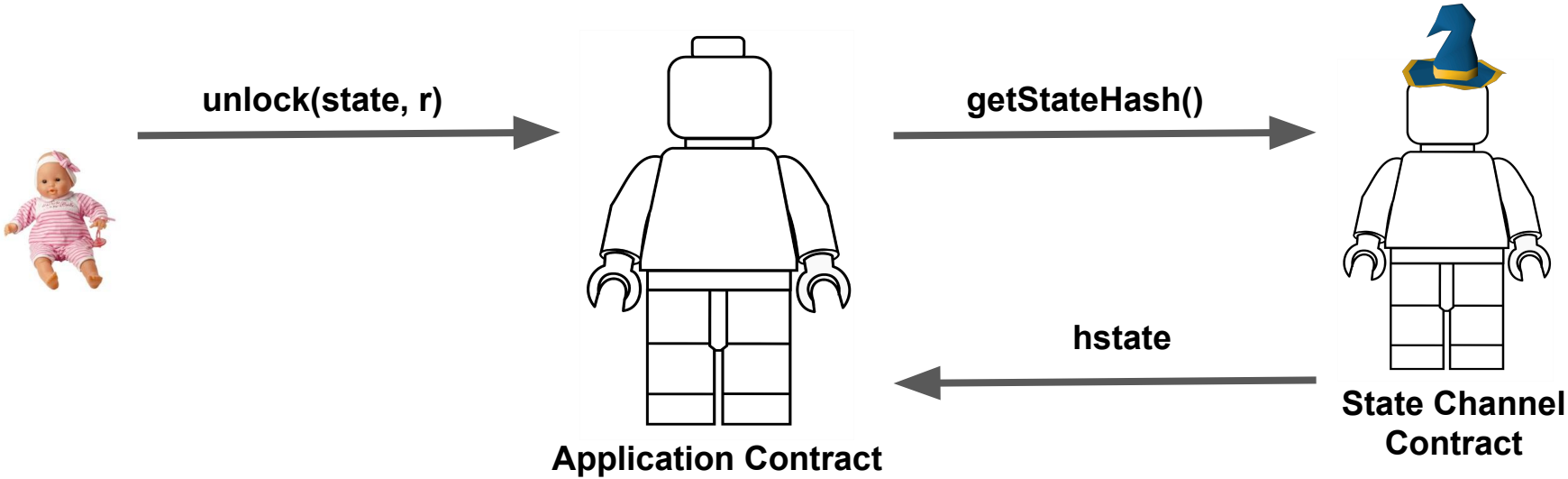
Step 1: Alice sends over full state (and blinding r)

Re-activation!



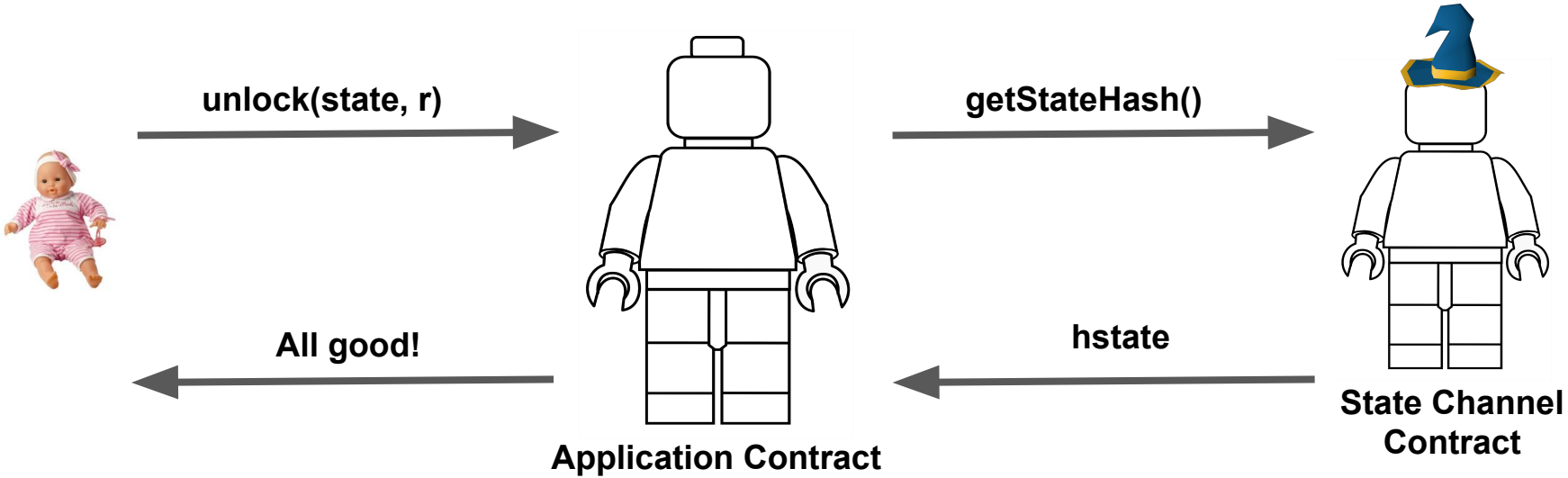
Step 2: AppCon fetches commitment to state (hstate) from state channel

Re-activation!



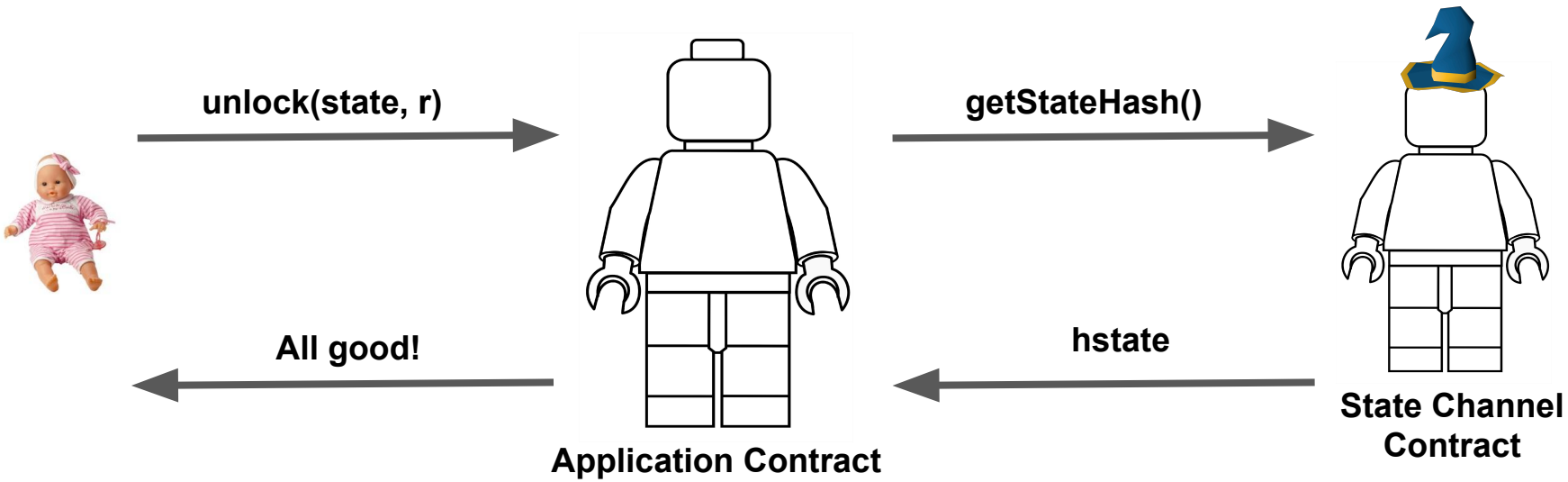
Step 3: AppCon checks that H(state, r) == hstate

Re-activation!



Step 4: AppCon stores state and re-activates all functionality

Re-activation!



Summary: Update all variables in the AppCon (i.e. the “state”) and re-enable all functionality to continue execution via the blockchain

Are there any new security assumptions?

Execution Fork

If Bob doesn’t come back online….

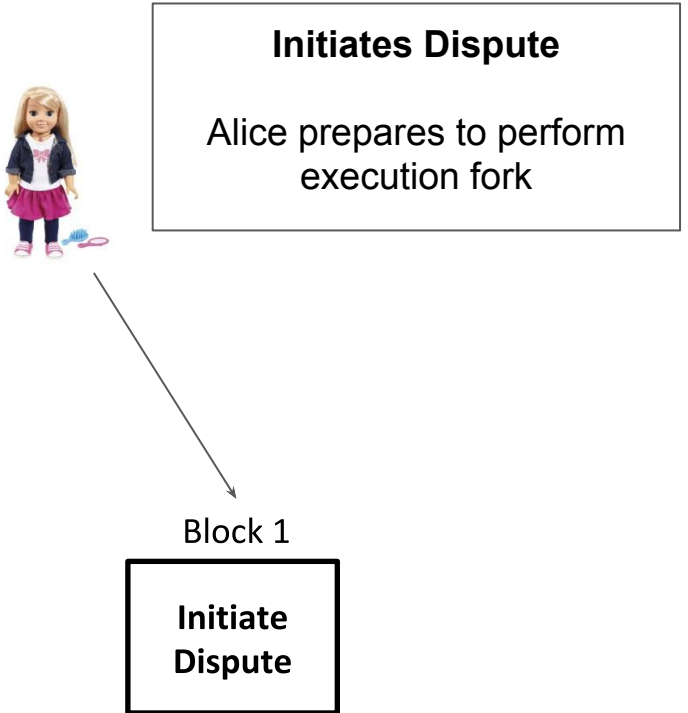
Alice will reverse the game and get her coins back!

Always online assumption:

All parties must remain online to detect and defend against execution forks

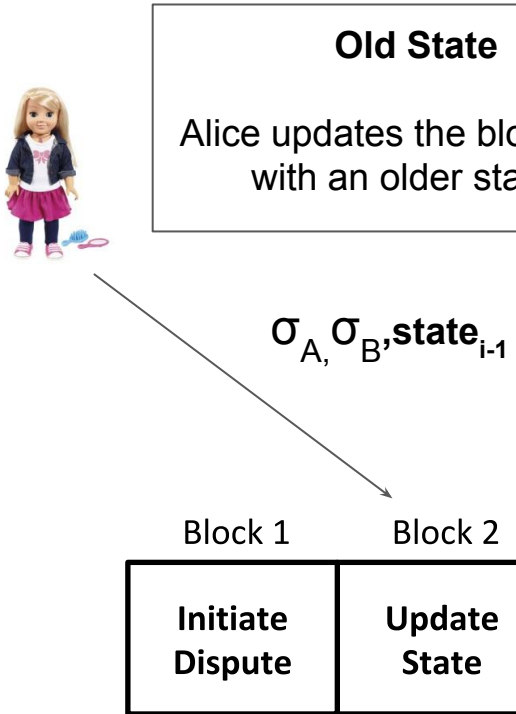
Always online assumption:

All parties must remain online to detect and defend against execution forks



Always online assumption:

All parties must remain online to detect and defend against execution forks



Always online assumption:

All parties must remain online to detect and defend against execution forks

Block 1	Block 2	Block 3
Initiate Dispute	**Update State**	

Always online assumption:

All parties must remain online to detect and defend against execution forks

AHHH OH NO

$\sigma_{A,}\sigma_{B}$,**state**$_{i}$

Block 1	Block 2	Block 3	Block 4
Initiate Dispute	**Update State**		**Update State**

Always online assumption:

All parties must remain online to detect and defend against execution forks

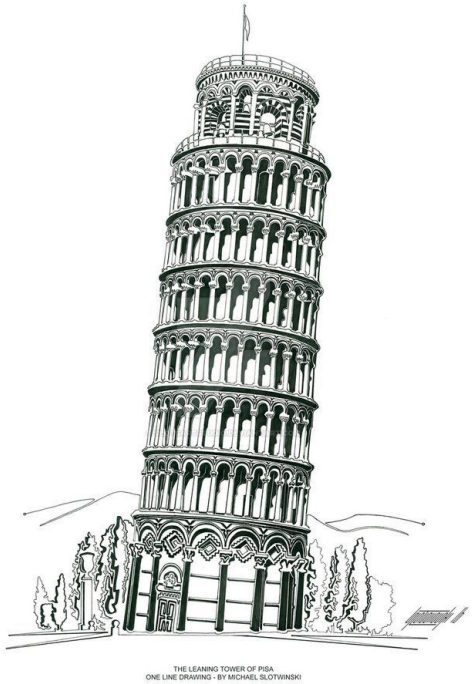
Execution Fork is Cancelled!

Bob will keep his winnings, but only because **he remained online and responded at the right time!**

Block 1	Block 2	Block 3	Block 4
Initiate Dispute	**Update State**		**Update State**

Can we help **alleviate** this new security requirement?

Pisa: Hire an accountable third party!



State Privacy

Custodian should not learn about the state he is watching (unless there is a dispute on-chain!)

Fair Exchange

Custodian should be paid upon accepting appointment from customer

O(1) Storage

Custodian only has to store the latest job received from the customer!

Recourse as Financial Deterrent

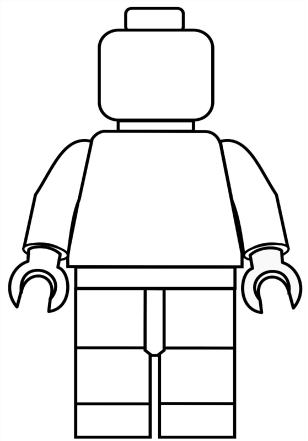
There should be indisputable evidence that can be used to punish a malice Custodian

Empirical Evaluation Thoughts

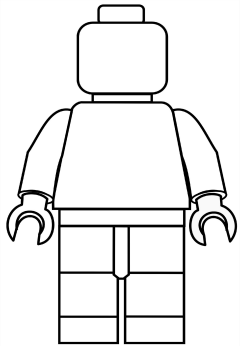
Application-Agnostic State Channels

Only handles hashes + signatures.

Works for any application.



Application Contract



State Channel Contract

Ideal case: No latency, no fees, two transactions on the blockchain

1 transaction to **turn on** channel

200+ transactions **authorised off-chain** for a single game of battleship

1 transaction to **turn off** channel

Funfair Dilemma

If the player is about to win a $10 bet, but the counterparty has stopped responding in the channel, then is it **worthwhile for the player** to:

Turn off the channel, complete the dispute process,
re-activate the application and win the bet via the blockchain

...if this process costs $100?

Worst case: Commit to playing game, turn off channel, play entire game on-chain.

1 transaction to **turn on** channel	2-3 off-chain transactions to **set up game**	1 transaction to **turn off** channel	200+ transactions **ALL on-chain** for a single game of battleship

Financial cost to resolve dispute

$1.56 to re-activate application and continue it via the blockchain

Transaction fee for every move

$16-24 to finish the full battleship game via the blockchain

Blockchain Latency is a killer

A grace period is required to let a transaction get accepted into the blockchain - for 5 mins - it takes several hours to complete a single game.

What applications make sense?

All parties remain online throughput the application's life-time

A **small group** of parties (due to lack of fault tolerance)

All parties want to **execute the application more than one time**

Examples: Payment, casino games, boardroom voting, auctions. etc

Ethereum Community Fund

Two years later.

ethereum foundation grants

We are on the verge of practical and deployable state channels.

Magic unicorns **need not apply.**

PISA:
http://hackingdistributed.com/2018/05/22/pisa/

Sprites:
https://arxiv.org/abs/1702.05812

Github for battleship evaluation:
https://github.com/stonecoldpat/statechannels

paddyucl