Brick
Asynchronous Payment Channels

Zeta Avarikioti
L. Kokoris-Kogias, R. Wattenhofer, D. Zindros

Fundamentals of Channels
Fundamentals of Channels

Funding transaction
Commitment transaction
Fundamentals of Channels

Commitment transaction

5 4

2 7

8 1
Fundamentals of Channels

Funding

Commitment

Dispute period
Inactive Counter Party

- Funding
- Commitment
- Dispute period
Attack the Liveness of the Blockchain

Funding
Commitment
Dispute period
Censorship Congestion
Time = CryptoMoney!
Time = CryptoMoney!

Asynchronous channels?
Be proactive, not reactive

I believe in a better way.
Be proactive, not reactive

Signatures of Alice & Bob
OR
Signatures of WT & (Alice or Bob)
Watchtower Committee

Committee
n = 3f+1
f Byzantine

Signatures of Alice & Bob
OR
Signatures of 2f+1 WTs & (Alice or Bob)
Challenges

1) Consensus is costly
2) Privacy is important
3) Incentives are critical
Consistent Broadcast

- O(n) communication complexity for state updates
- Verification of consensus between Alice & Bob
- No guarantees, if Alice & Bob both misbehave
Privacy

- Privacy preserving
- Alice/Bob cannot publish a previous transaction
Brick Architecture

(1) Update

(2) Consistent Broadcast

(3) Execute

H( )

counter

Close: max state of 2f+1 submitted states.
Safety
A channel will only close in the freshest committed state

Brick Security Analysis

f slow honest WTs

2f+1 WTs closing state (previous committed state)

2f+1 WTs freshest committed state
Brick Security Analysis

Liveness
Any valid operation (close, update) will eventually be committed

Not committed = Invalid operation (failed verification)
Challenges

1) Consensus is costly
2) Privacy is important
3) Incentives are critical
Why be a Watchtower?
Per-update fees

Repeated game lifts the fair-exchange impossibility
Watchtower paid while channel is alive!
Incentives to close?
Why assist to close honestly?

Collateral
Why assist to close honestly?

Collateral

Asynchronous channels?
Fraud proofs

two signed conflicting states

Party claims the collateral
Fraud proofs
two signed conflicting states

Party claims the collateral

channel value $v$

claimed collateral $\frac{v}{f} \times (f+1)$
Collateral

Where do we close?
when \( >f \) fraud proofs are submitted

all channel value→ counterparty
Where do we close?
when $\leq f$ fraud proofs are submitted

run close again without the malicious $\rightarrow$ max state of $2f+1$
Collateral

Profit =
channel balance (c) + fraud proofs (v/f) - bribes (v/f + \varepsilon)

v = channel value
f = Byzantine watchtowers
y = bribed watchtowers
Collateral

Profit =
channel balance (c) + fraud proofs (v/f) - bribes (v/f + ε)

1. 0 FPs: profit = c ≤ v

v = channel value
f = Byzantine watchtowers
y = bribed watchtowers
Collateral

Profit =
channel balance (c) + fraud proofs $\frac{v}{f}$ - bribes $\frac{v}{f} + \varepsilon$

1. 0 FPs: profit $= c \leq v$

2. $> f$ FPs: profit $\leq v + y \frac{v}{f} - y \left(\frac{v}{f} - \varepsilon\right) = v - \varepsilon$

$v$ = channel value
$f$ = Byzantine watchtowers
$y$ = bribed watchtowers
Collateral

Profit = channel balance \((c)\) + fraud proofs \((v/f)\) - bribes \((v/f + \varepsilon)\)

1. **0 FPs**: profit = \(c \leq v\)

2. **> f FPs**: profit \(\leq v + y^*\frac{v}{f} - y^*(\frac{v}{f} - \varepsilon) = v - \varepsilon\)

3. **f FPs and “correct” close**: profit = \(c + v\)

\(v = \) channel value  
\(f = \) Byzantine watchtowers  
\(y = \) bribed watchtowers
Collateral

Will a party close in a “incorrect” state?

<table>
<thead>
<tr>
<th>Action</th>
<th>Proof-of-fraud</th>
<th>Close</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byzantine</td>
<td>m</td>
<td>$f - m$</td>
<td>$f$</td>
</tr>
<tr>
<td>Bribed</td>
<td>y</td>
<td>$f + 1 - (f - m)$</td>
<td>$y + m + 1$</td>
</tr>
<tr>
<td>(rational)</td>
<td></td>
<td>$= m + 1$</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>$m + y$</td>
<td>$f + 1$</td>
<td></td>
</tr>
</tbody>
</table>

profit = $\nu + \frac{(m+y)\nu}{f} - (y+m+1)\left(\frac{\nu}{f} + \varepsilon\right) \leq \nu - \frac{\nu}{f} - \varepsilon < c + \nu$

channel value fraud proofs bribes
Collateral

Profit =
channel balance \((c)\) + fraud proofs \((\frac{v}{f})\) - bribes \((\frac{v}{f} + \varepsilon)\)

1. **0 FPs**: profit = \(c \leq v\)

2. **> f FPs**: profit \(\leq v + y*\frac{v}{f} - y*(\frac{v}{f}-\varepsilon) = v - \varepsilon\)

3. **f FPs and “correct” close**: profit = \(c + v\)

4. **f FPs and “incorrect” close**: profit = \(v - \frac{v}{f} - \varepsilon\)

\(v = \text{channel value}\)
\(f = \text{Byzantine watchtowers}\)
\(y = \text{bribed watchtowers}\)
Why assist to close?

WTs collude → Hostage situations

Closing fees
prisoner’s dilemma
Committee size $> 7$
richest party loses more

Parties collude $\rightarrow$ Hostage situations
The more (WTs) the merrier!

↑ robustness
↓ collateral per WT
≃ cost for parties
Brick Cost

Gas cost of a BRICK channel

- Cost of deployment
- Cost of opening
- Cost of pessimistic close
- Cost of optimistic close
- Recommended n

EUR

Ether

Number $n$ of WARDENS

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07
Brick Advantages

- Privacy
- Incentive-compatible
- Good performance
- Asynchronous
  - censorship
  - congestion
  - liveness attacks
Limitations, Extensions & Future Work

- Minimum collateral
- Update fees via one-way channel
Limitations, Extensions & Future Work

- Minimum collateral
- Update fees via one-way channel
- Watchtower replacement
- Consensus → fork resilient
- Auditability
Brick+
Brick+ Workflow

(1) On-chain Audit Request
On-chain Audit Request

Committee Closes the Channel
Brick++ Workflow

1. On-chain Audit Request
2. Committee Closes the Channel
3. Parties transfer the state history to the auditor

Funding  Audit  Close  Last hash

(3)Transfer(S₁, S₂, ..., Sₙ)
(1) Audit
(2) Close
On-chain Audit Request
Committee Closes the Channel
Parties transfer the state history to the auditor
Committee publishes $H_s$
Brick+ Workflow

(1) On-chain Audit Request
(2) Committee Closes the Channel
(3) Parties transfer the state history to the auditor
(4) Committee publishes $H_s$
(5) Auditor verifies $(H_s, S_1, S_2, \ldots, S_n)$
Brick+ Workflow

1. On-chain Audit Request
2. Committee Closes the Channel
3. Parties transfer the state history to the auditor
4. Committee publishes $H_s$
5. Auditor verifies $(H_s, S_1, S_2, ..., S_n)$
Limitations, Extensions & Future Work

- Minimum collateral
- Update fees via one-way channel
- Watchtower replacement
- Consensus $\rightarrow$ fork resilient
- Auditability
- Multiple parties
Thank you!

Questions?