Plasma Cash
Towards improved Plasma constructions

Georgios Konstantopoulos / @gakonst
Table of Contents

1. Overview
2. Transfers, Exits and Challenges
3. Security and Incentive Compatibility
4. Future / Ongoing Work
5. Summary
Overview
Relative Work

Layer 2 Design Space - Blockchain “connected” to Layer 1

1. Sidechains
   a. Two-way pegged - Merged Mined (SPV Proofs / NIPoPoWs)
   b. Federated peg (multisig)
2. Drivechains
3. Shadowchains
4. Treechains
5. Client-Side Validation
6. NOCUST
Plasma = framework for “non-custodial” sidechains
What is a sidechain?

- Lock ETH in escrow (smart contract)
- Plasma-ETH transfer
- Mint Plasma-ETH
- Burn Plasma-ETH
- ETH is unlocked
Each Plasma Block Root is committed to the parent chain

submitBlock(0x...)

[Diagram of Plasma Block Roots and the Ethereum logo]
Each Plasma Block Root is committed to the parent chain.
Exit Game: Delayed Withdrawals

Unlock funds by interacting with parent-chain smart contract
Exit Game: Delayed Withdrawals

Challenged exits get cancelled
Non-Fungible Plasma, aka Plasma Cash

- UTXO ID: Leaf index in *Sparse Merkle Tree*
- Deposit → Receive coin with serial number (like cash!)
- 1 input - 1 output UTXOs
- Transact: Reference “parent” tx
  - 2 txs with same parent → double spend
- Exit: Reveal tx + parent tx

Deposit 5 ETH → Smart contract

Ethereum Smart contract

emit Deposit(serialNo)

Plasma Chain

Create Plasma Block
What’s a (Sparse) Merkle Tree?

H( H( H(A) + H(∅) ) + H( H(∅) + H(D) ) )

H( H(A) + H(∅) )

H(A)

A

H(∅)

∅

H(∅) + H(D)

H(∅)

∅

H(D)

D

https://medium.com/@kelvinfichter/whats-a-sparse-merkle-tree-acda70aeb837
Inclusion Proof

\[ H( H( H(A) + H(B) ) + H( H(C) + H(D) ) ) \]
Exclusion Proof

\[ H( H( H(A) + H(\emptyset) ) + H( H(\emptyset) + H(D) ) ) \]
Transfers, Exits & Challenges
Example of Coin Transfer (Deposit 5 ETH)

Deposit 5 ETH on Plasma Contract, 5 ETH NFT appears on the Plasmachain “from nowhere”
Example of Coin Transfer (Deposit 5 ETH)

Alice sends coin to Bob + inclusion proof in Block 1. 
Bob must verify that the coin is valid.
Example of Coin Transfer (Deposit 5 ETH)

Block 3 gets submitted, the coin was not moved.
Example of Coin Transfer (Deposit 5 ETH)

Bob sends the coin to Charlie: Inclusion in blocks 1, 2 & exclusion in block 3
Charlie must verify the UTXO history since the coin’s deposit.
Example of Coin Transfer (Deposit 5 ETH)

Charlie verifies that the coin was valid. He then exits it by referencing an ancestor.
Each coin is its own unique state machine!
Happy Case
Non-Interactive Challenge
Interactive Challenge

Diagram:

- Start at $q_a$
- Transition to $q_b$ via $E$
- Transition to $q_c$ via $F$
- Transition to $q_d$ via $W$
- Transition back to $q_a$ via $C$
- Transition back to $q_b$ via $R$
- Transition back to $q_c$ via $IC$
Interactive Challenge → No Response
Interactive Challenge → All challenges responded
Exiting a spent coin?
Double Spending a Coin?
Exiting a coin with invalid history?
Exiting a coin with invalid history?
Security & Incentive Compatibility
Security of the Exit Game

\[ t_1 + D < t_0 + T \rightarrow \text{attack cancelled} \]

- \( t_0 \): Malicious Exit
- \( t_1 \): Challenge broadcast
- \( t_1 + D \): Challenge included
- \( t_0 + T \): Attack cancelled
Security of the Exit Game

\[ t_1 + D < t_0 + T \rightarrow \text{attack cancelled} \]
Security of the Exit Game

Malicious Exit
Challenge broadcast

$t_1 + D > t_0 + T \rightarrow \text{attack succeeds}$
Security of the Exit Game

Malicious Exit

Challenge broadcast

Challenge included

Safety condition: \( D \leq T + t_0 - t_1 \)

\( t_1 + D > t_0 + T \rightarrow \text{attack succeeds} \)
Attacker Decision Flow

Malicious Exit

Challenged

- Attack Failed

No challenge

- Attack Succeeds

  + Full bond refunded
  + Coin value obtained
  - Exit Gas

Pay gas + bond
Attacker Decision Flow

Malicious Exit

- Pay gas + bond

Challenged

- Frontrun
  - Losses cut
    - a% of bond refunded
    - Exit Gas
    - Challenge Gas
  - Frontrun fails
    - Big losses
      - 100% of bond lost
      - Exit Gas
      - Challenge Gas

No challenge

- Attack Succeeds
  + Full bond refunded
  + Coin value obtained
  - Exit Gas
Incentive Compatibility of the Exit Game

\[ E(R) = P(C)v \leq 0 \]

No challenges = success:
- \( \uparrow \) onchain congestion / censorship
- \( \uparrow \uparrow \) block withholding
- \( \downarrow \) liveness of participants
- \( \downarrow \) challenge period \( T \)

Large \( T \) = Secure but bad UX!
Incentive Compatibility of the Exit Game

\[ E(R) = P(C)\nu - \left[ \text{gas} + P(C) \times \text{bond} \right] \]

Cost to Attack =
- Gas Costs (constant)
- **Fidelity Bond** (goes to challenger)

No challenges = success:
- \( \uparrow \) onchain congestion / censorship
- \( \uparrow \) block withholding
- \( \downarrow \) liveness of participants
- \( \downarrow \) **challenge period** \( T \)

Large \( T \) = Secure but bad UX!
Incentive Compatibility of the Exit Game

\[ E(R) = P(\bar{C})v - [\text{gas} + P(C) \cdot \text{bond}] + P(C)P(F \mid C) \cdot \text{bond} \leq 0 \]

Cost to Attack =
- Gas Costs (constant)
- **Fidelity Bond** (goes to challenger)

Frontrunning removes bond from cost if successful

No challenges = success:
- \(\uparrow\) onchain congestion / censorship
- \(\uparrow\) block withholding
- \(\downarrow\) liveness of participants
- \(\downarrow\) challenge period \(T\)

Large \(T\) = Secure but bad UX!

\[ P(F \mid \bar{C}) = 0 \]

Attacker won’t frontrun if nobody challenged
Incentive Compatibility of the Exit Game

\[ E(R) = P(C) v - [\text{Gas Costs} + \alpha P(C) P(F | C) \times \text{bond}] \]

Cost to Attack =
- Gas Costs (constant)
- Fidelity Bond (goes to challenger)

No challenges = success:
- \(\uparrow\) onchain congestion / censorship
- \(\uparrow\) block withholding
- \(\downarrow\) liveness of participants
- \(\downarrow\) challenge period \(T\)

Large \(T\) = Secure but bad UX!

Frontrunning removes bond from cost if successful

\[ P(F | C) = 0 \]

Attacker won’t frontrun if nobody challenged

Burn part of the bond.
Future / Ongoing Work
Arbitrary Denomination Payments

Non-fungible coins → double edged-sword

- Change providers: Pay with 7 ETH for a 5 ETH product, get a 2 ETH coin in return. Needs to be atomic
- Plasma Debit → Each coin can have value between 0 and its max capacity (~payment channels on Plasma)
- Initially break a coin in small pieces (Plasma Cash Defragmentation/”Cashflow”)
Reduce data requirements for light clients

Problem: Linearly increasing proof size of coin history.

- Checkpoints: Periodically being able to discard history older than some time (Plasma XT)
- Less frequent commitments to the main network less often (without losing finality and/or throughput)
- Accumulators / Vector Commitments
- zkSNARK/STARKs for compression
State Channels + Plasma?

Benefits:

- Smart Contracts on Plasma!
- Fast & fee-less opening/cooperative closing of channels
- 0 cost to update channel parameters

Requirements:

- Multisig accounts for escrow
- Timelocked UTXOs for non-cooperative cases
Summary

- Non-custodial sidechain via notarization of blocks
- Off-chain gas-less fixed denomination payments with mainchain finality - no onboarding cost
- “Compression” mechanism (more txs settle per block)
- Users must audit MAINCHAIN contract for fraud (*light* client side validation)

WIP:
- Smart Contracts
- Arbitrary Denomination Payments
- *Even lighter* light clients
Smart Contracts via State Channels

Alice & Bob transfer their coins to a multisig
Smart Contracts via State Channels

Alice & Bob transfer their coins to a multisig

...and start a Tic Tac Toe game

Tic Tac Toe Channel
Smart Contracts via State Channels

Alice & Bob transfer their coins to a multisig

Tic Tac Toe Channel

Game Winner: Alice

Cooperative close
What if Bob goes offline?

- Alice & Bob transfer their coins to a multisig

- Bob’s offline (or refuses to sign)

- Unilateral settlement

Game Winner: Alice
Thank you for your attention

@gakonst / me@gakonst.com
github.com/loomnetwork/{plasma-paper, plasma-cash}