Non Custodial Sidechains for Bitcoin utilizing Plasma Cash and Covenants

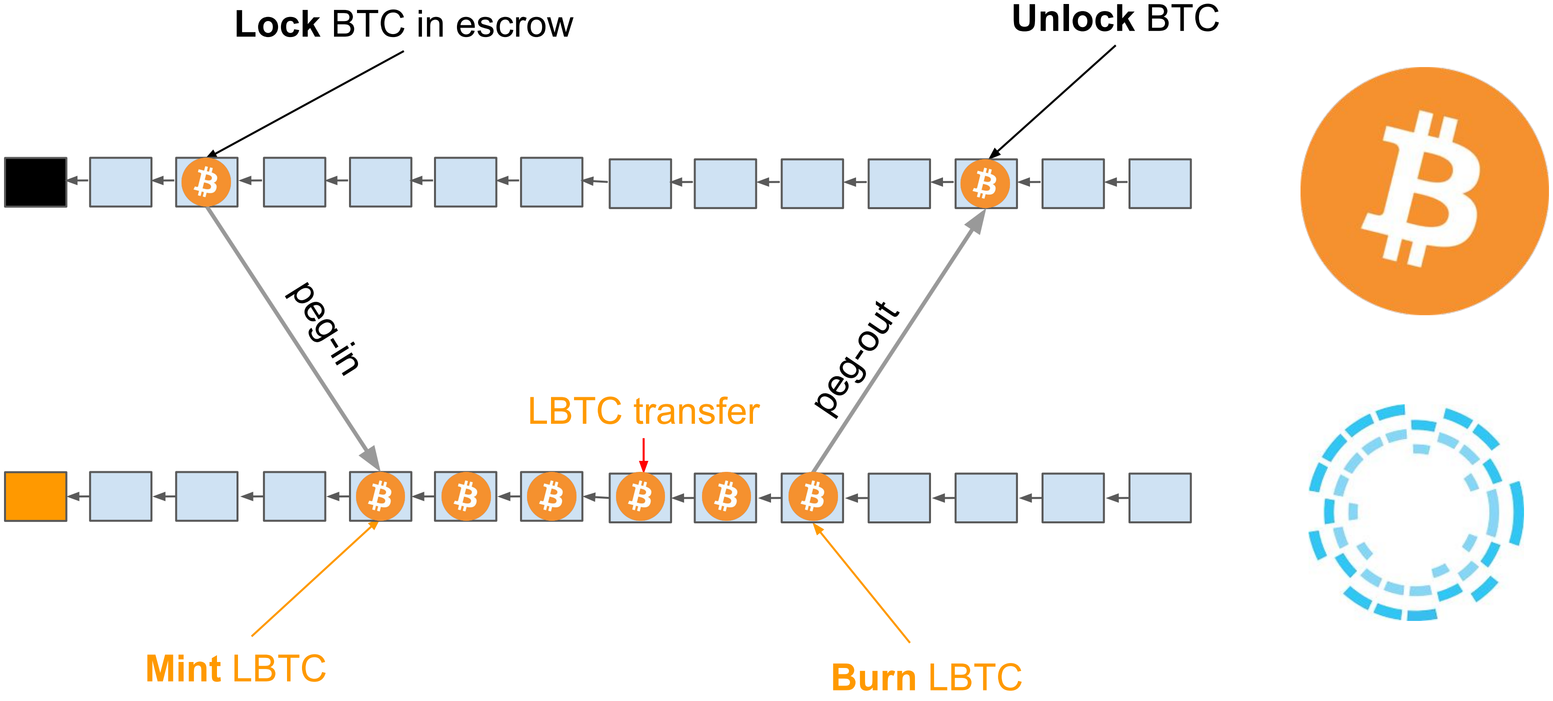
Georgios Konstantopoulos
Independent Consultant & Software Engineer
Twitter: @gakonst / me@gakonst.com

Slides available: gakonst.com/scalingbitcoin2019.pdf

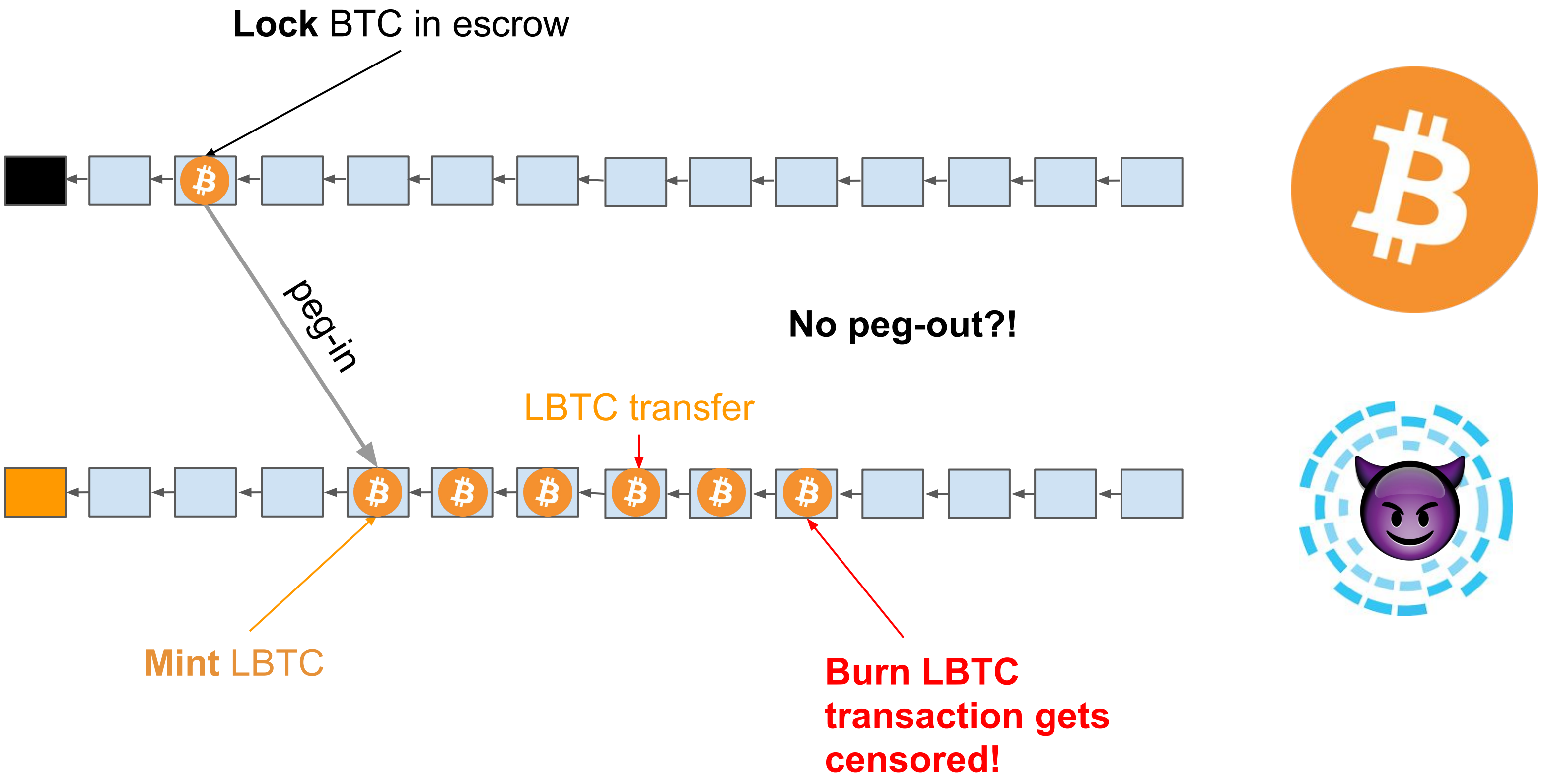
How do we scale?

1. Increase semantic density of transactions
   (Segwit / MAST / Schnorr / Taproot / … / **Layer 2**)
2. ~~Bigger blocks~~

Sidechains considered harmful



Sidechains considered harmful



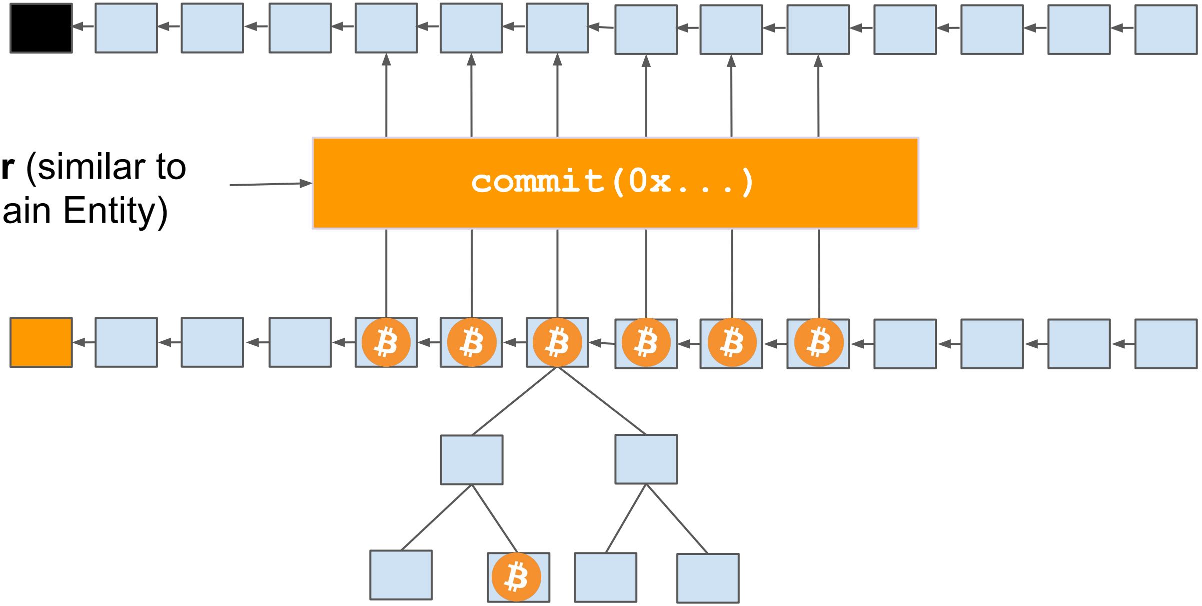
Plasma Cash in 1 slide

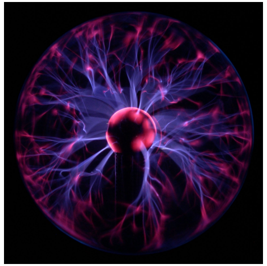
- Untrusted operator notarizes off-chain state
- Operator can censor but cannot steal
- Fixed denomination transfers like cash
- Safe under liveness assumption (~2KB stale state fraud proof)
- Watchtower compatible
- No overcollateralization requirements
- No need to sign to receive a transfer
- Can receive funds without on-chain transaction (no notion of inbound liquidity)

https://docs.google.com/spreadsheets/d/1vHw3Rr7aFH1FXIyMjnhC0Wvro_IZ4S8hozpaWgMR1bc/

“Operator” commits* each block root to “parent chain”

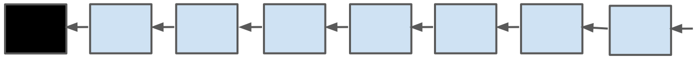
Operator (similar to Statechain Entity)

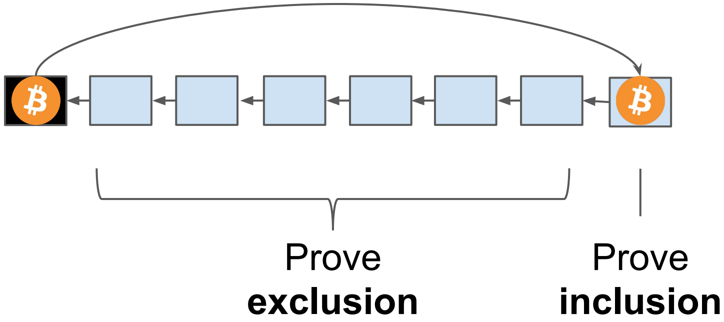


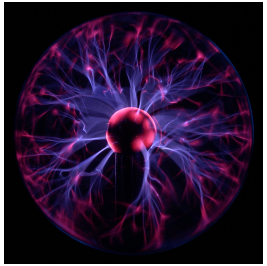


*uses accumulator that supports non-membership proofs e.g. ordered merkle tree

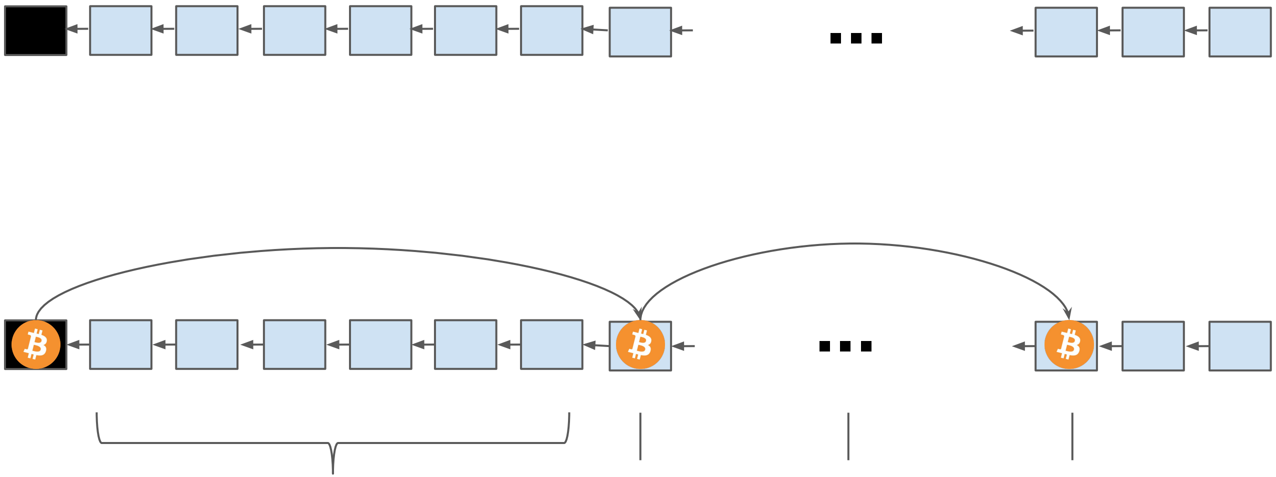
Prove coin history per transfer (off-chain)







Prove coin history per transfer (off-chain)

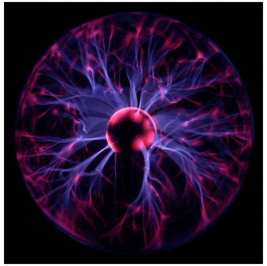


Prove **exclusion**

Prove **inclusion**

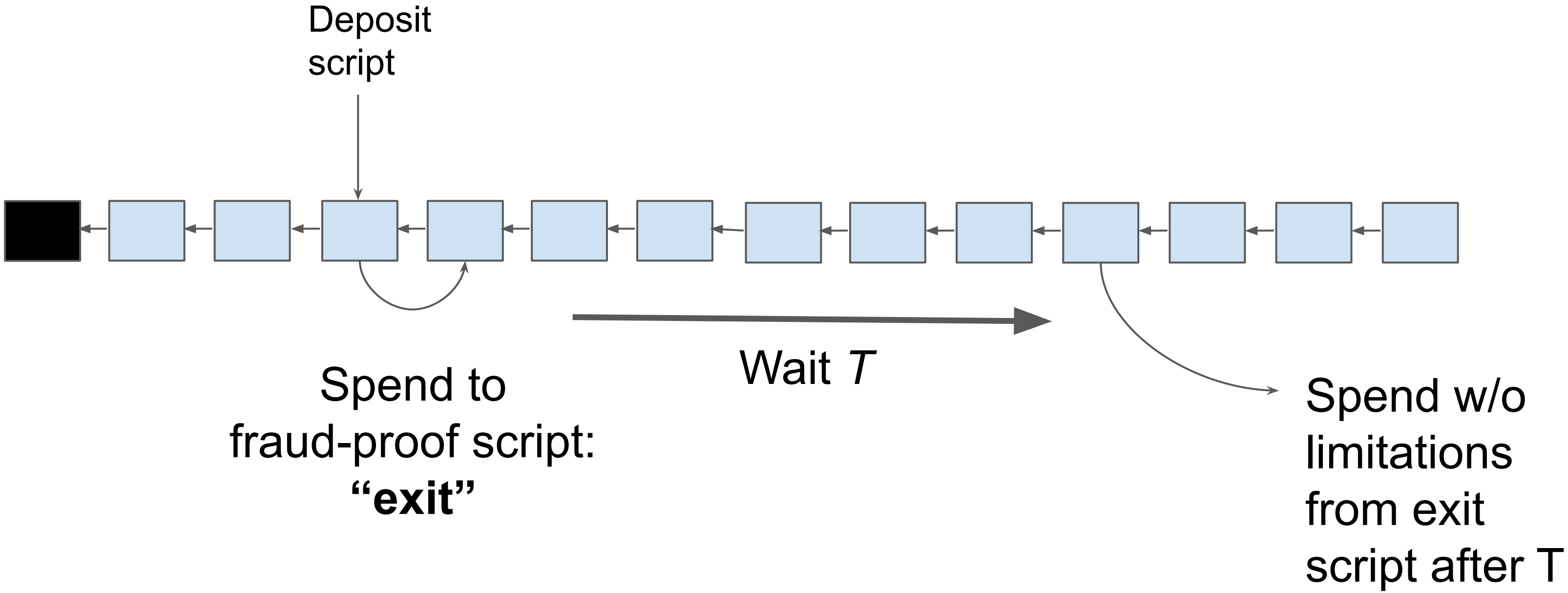
Prove **exclusion**

Prove **inclusion**



Coin history grows linearly with number of blocks
TXO Commitments? RSA Accumulators?

Exit Game: Delayed Withdrawals



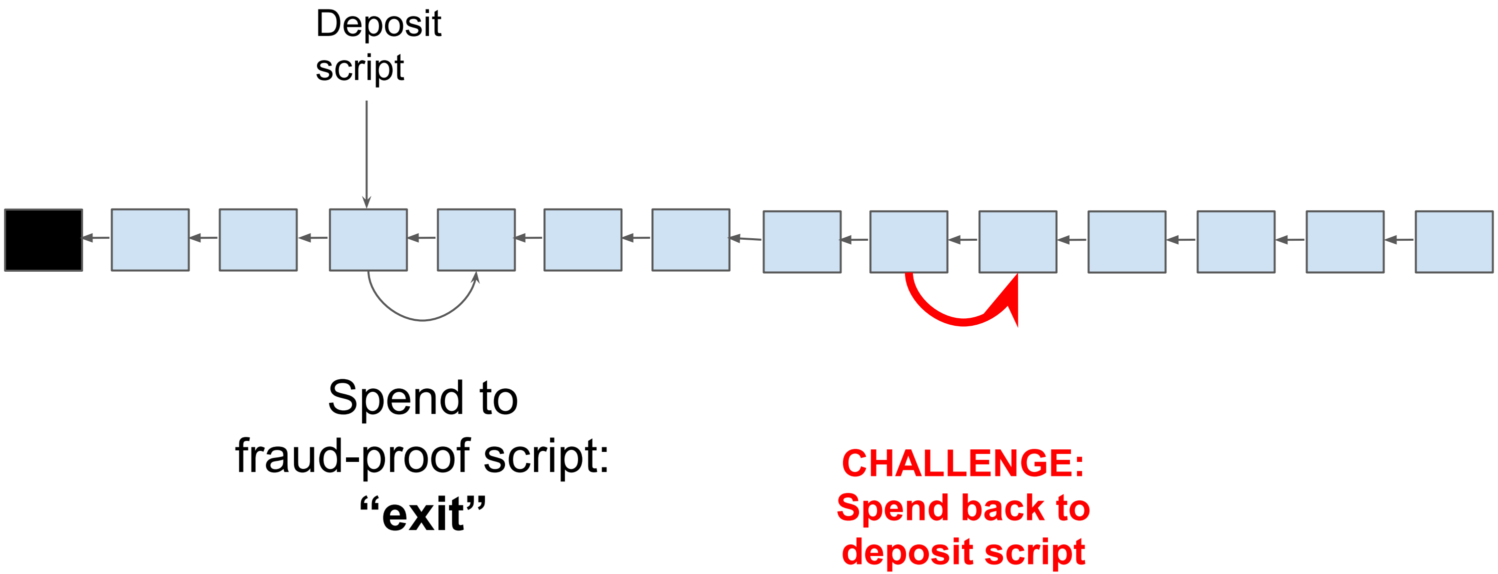
Deposit script

Spend to fraud-proof script: **“exit”**

Wait *T*

Spend w/o limitations from exit script after T

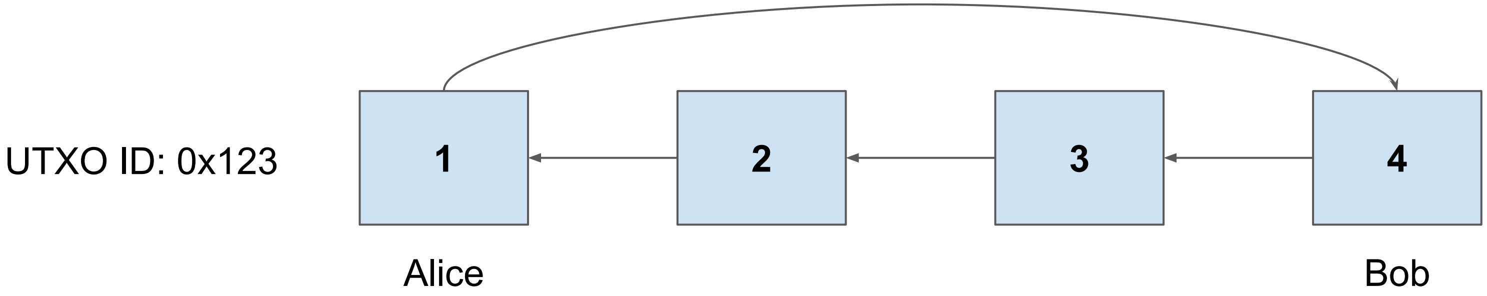
Exit Game: Delayed Withdrawals


Deposit script
Spend to fraud-proof script: **“exit”**
CHALLENGE: Spend back to deposit script

Transaction Format: 1 input 1 output UTXO

(UTXO_ID, PARENT_BLOCK, NEW_OWNER, PREV_OWNER_SIG)

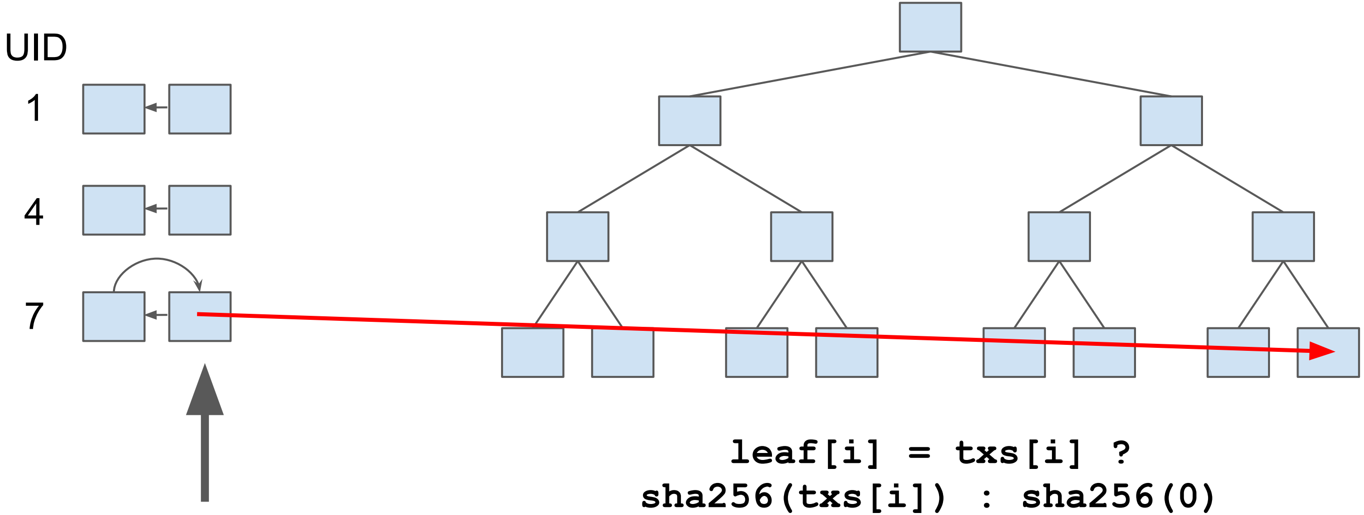
(0x123, 1, Bob, Alice_sig)



https://ethresear.ch/t/plasma-cash-was-a-transaction-format/4261

Merkle Tree: TxHash at each UTXO_ID index

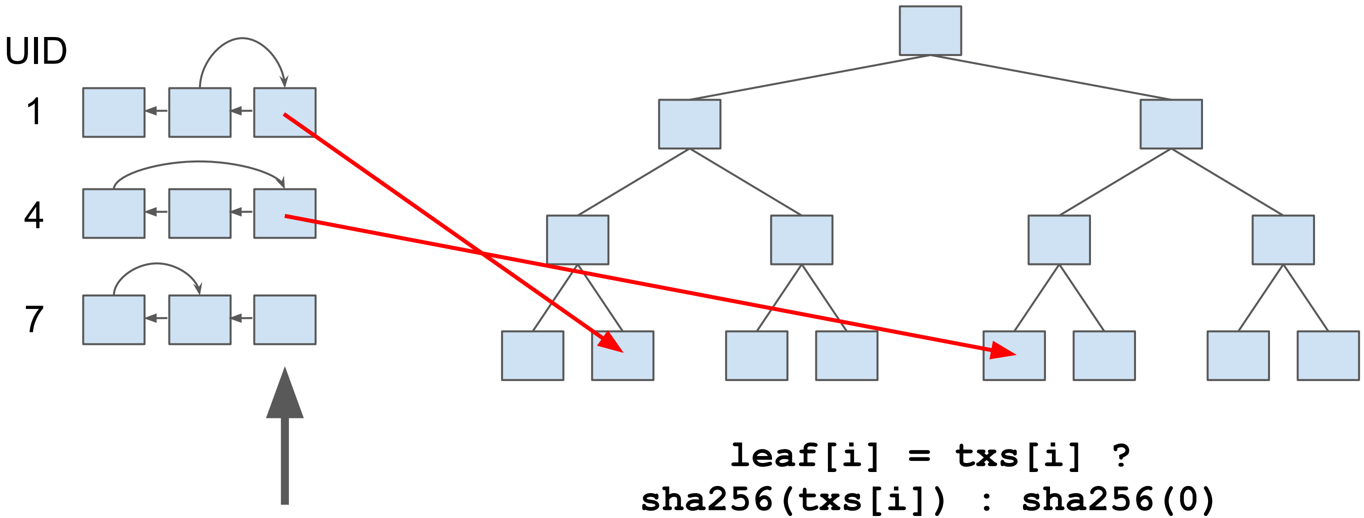
Current Block: 2



```
leaf[i] = txs[i] ?
sha256(txs[i]) : sha256(0)
```

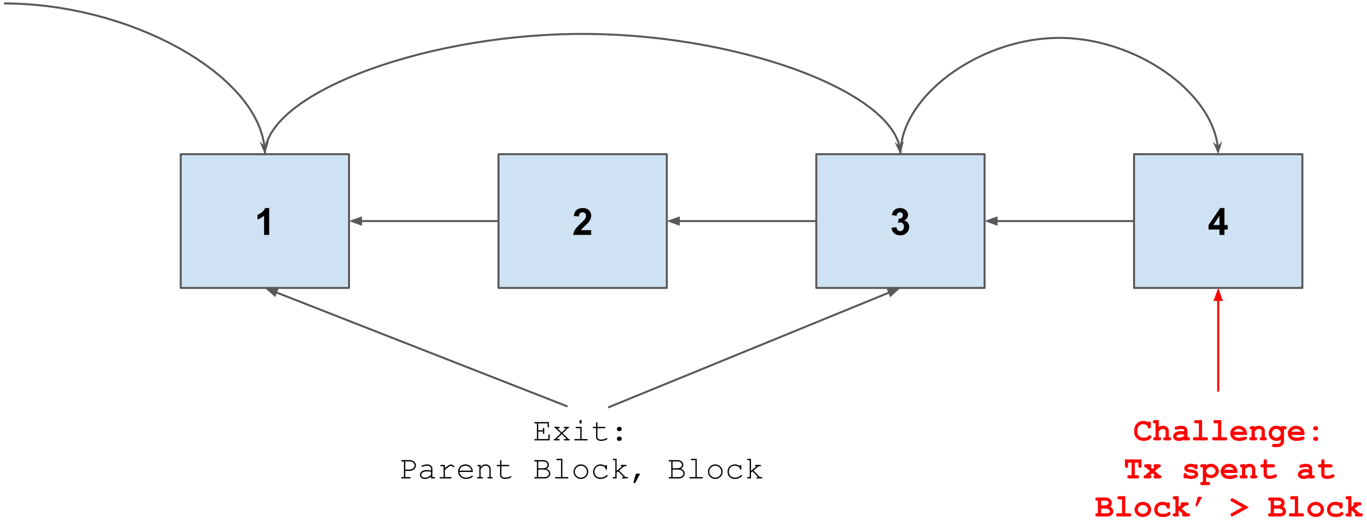
Merkle Tree: TxHash at each UTXO_ID index

Current Block: 3

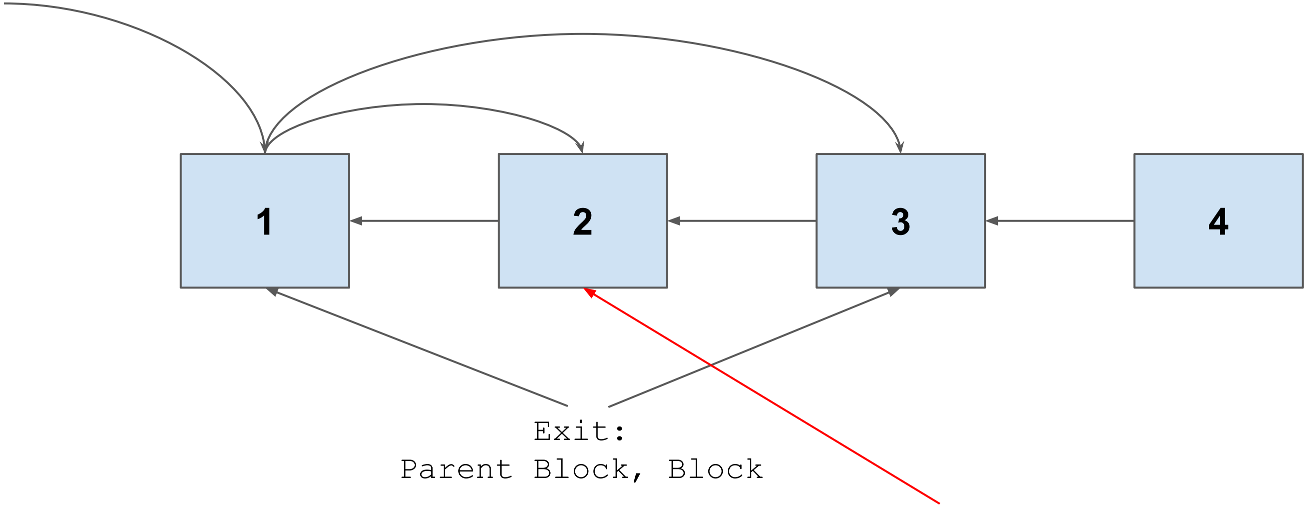


```
leaf[i] = txs[i] ?
sha256(txs[i]) : sha256(0)
```

“Exit Spent Coin”



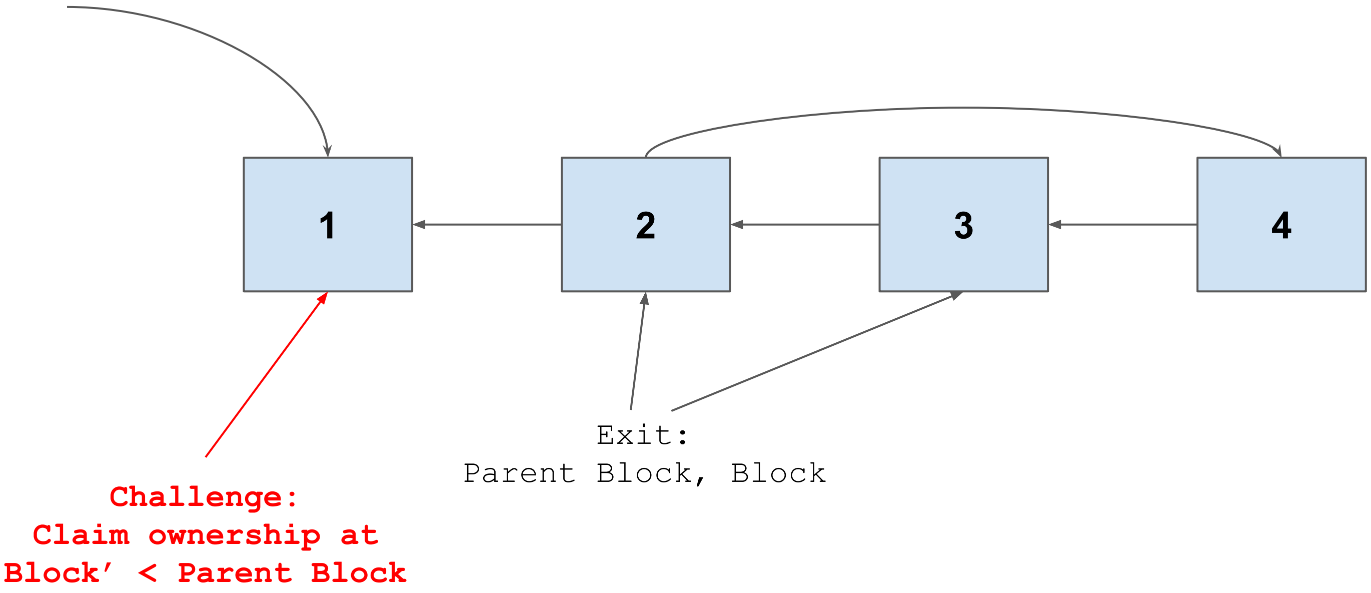
“Exit Double Spend” (malicious operator)



Exit:
Parent Block, Block

Challenge:
Parent Tx spent at
Parent Block < Block’ < Block

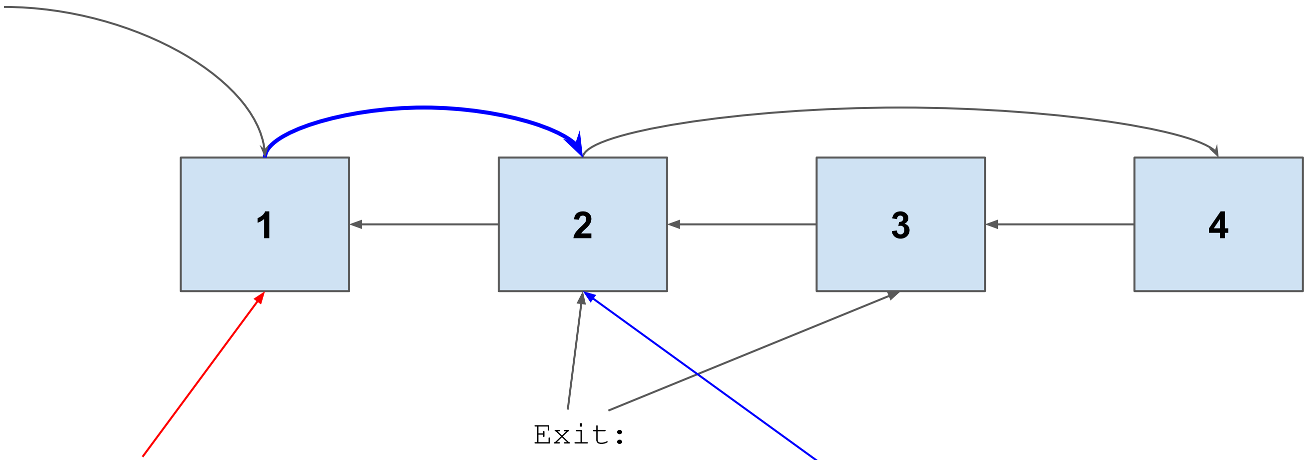
“Invalid History Challenge” (malicious operator)



Exit:
Parent Block, Block

Challenge:
Claim ownership at
Block’ < Parent Block

Response to Invalid History Challenge

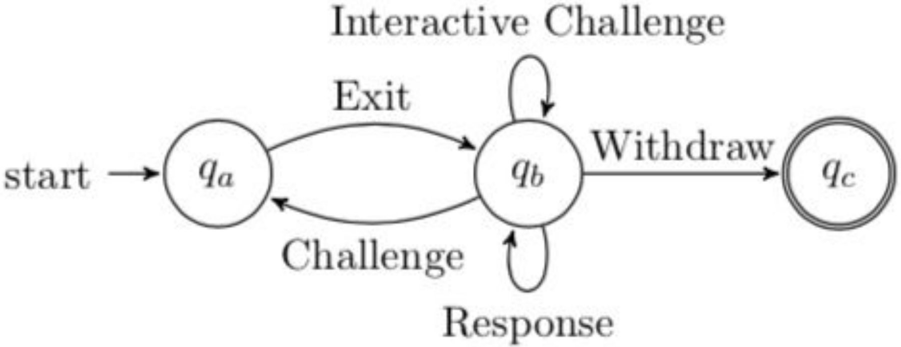


Exit:
Parent Block, Block

Challenge:
Claim ownership at
Block’ < Parent Block

Response:
Reveal spend from 1 at
Block’ < Block’’ <= Parent Block

Exit Game state machine (per coin)



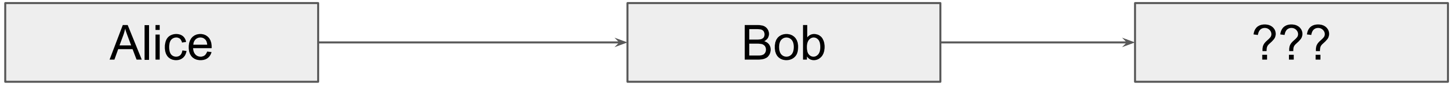
Background literature on covenants

What is a covenant?

Restriction on the outputs spending a UTXO.

O’Connor @ Bitcoin Workshop 2017:

- Digital signatures: **WHO** can spend Bitcoin
- Timelocks: **WHEN** Bitcoin can be spent

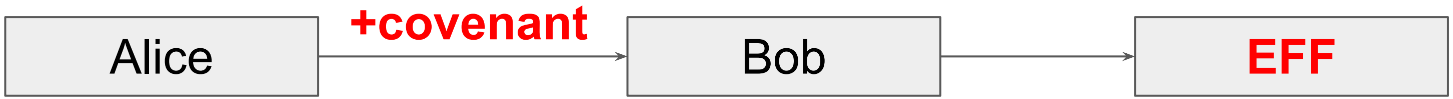


What is a covenant?

Restriction on the outputs spending a UTXO.

O'Connor @ Bitcoin Workshop 2017:

- Digital signatures: **WHO** can spend Bitcoin
- Timelocks: **WHEN** Bitcoin can be spent
- Covenants: **HOW** and **WHERE** Bitcoin can be spent



Use Cases

- Vaults
- Paralysis Proofs
- Colored Coins (non-fungible tokens)
- Congestion Control
- **Fraud proofs → Sidechains with trust-minimized reverse peg**
- ...more in the mailing list

Covenant Designs

- OP_CHECKOUTPUT (MES’16)
- OP_CAT + OP_CHECKSIGFROMSTACK (O’Connor, Piekarska ‘17)
- OP_CHECKOUTPUTSHASH / OP_SECURETHEBAG (Rubin)
- OP_PUSHTXDATA (Lau)
- Presigned Transactions (McElrath / Bishop)

Implementing
Plasma Cash on Bitcoin

Merkle Proof Verification

```
VerifyIncluded(UTXO_ID, ROOT, TX_HASH, PROOF):

    ROOT
    TX_HASH
    PROOF
    UTXO_ID
    MERKLEBRANCHVERIFY
```

Verify block root was signed by Operator

```
VerifySignedByOperator(BLOCK_NUM, ROOT, SIG):

    BLOCK_NUM
    ROOT
    CAT
    SIG
    <OPERATOR_ADDRESS>
    CHECKSIGFROMSTACKVERIFY
```

Verify transaction was signed by previous owner

```
VerifyTxSigned(tx)

    UTXO_ID
    PARENT_BLOCK_NUM
    NEW_OWNER
    CAT CAT SHA256
    SIG
    <PREV_OWNER_ADDRESS>
    CHECKSIGFROMSTACKVERIFY
```

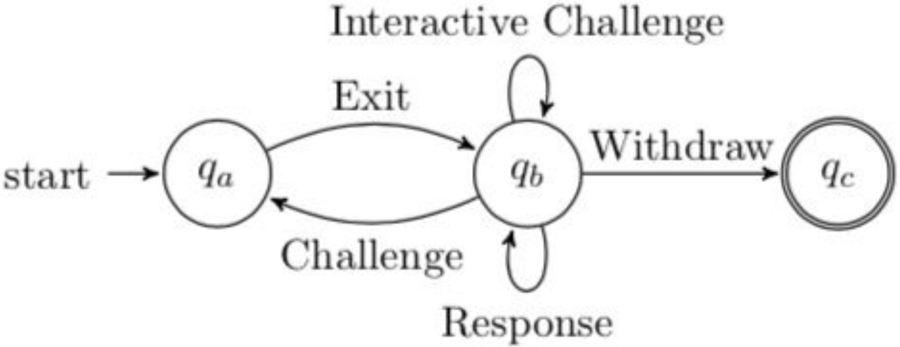
Enforce UTXO is spent to next state

```
EnforceSpentTo(ARGS, NEXT_STATE_PATTERN):
    ARGS
    NEXT_STATE_PATTERN
    CHECKOUTPUTVERIFY
```

(use `PICK` etc. to put `<ARGS>` in `OUTPUT_PATTERN` / dynamically construct the covenant during redemption)

Putting it all together

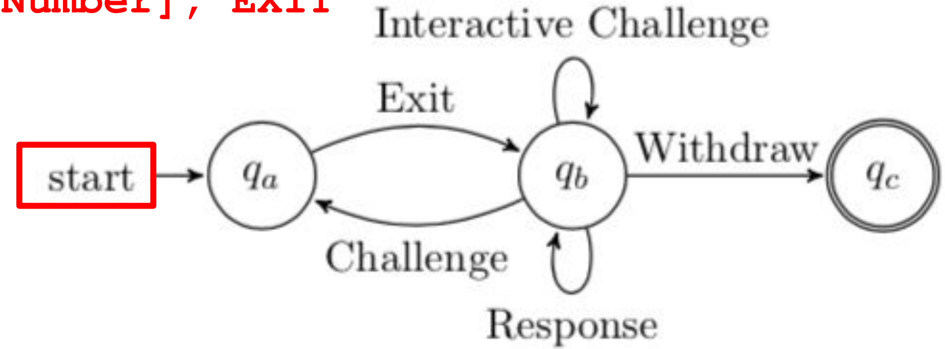
(verify tx merkle proofs, operator & transaction signatures w/ previous scripts)



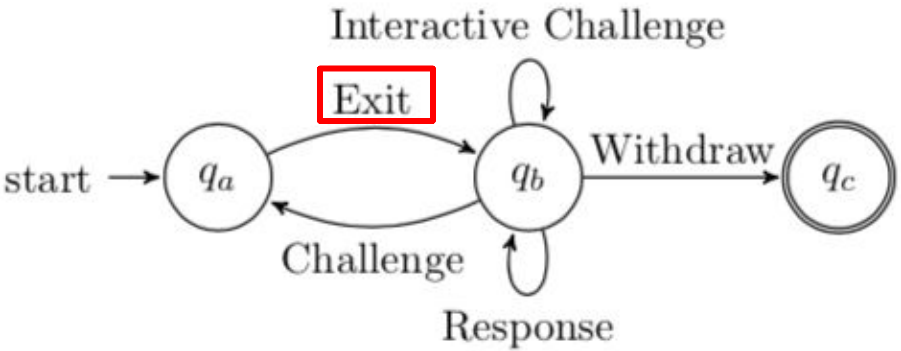
Deposit = Spend to covenant

Spend to `EnforceSpentTo(`

```
  [SidechainBlockNumber], EXIT
)
```

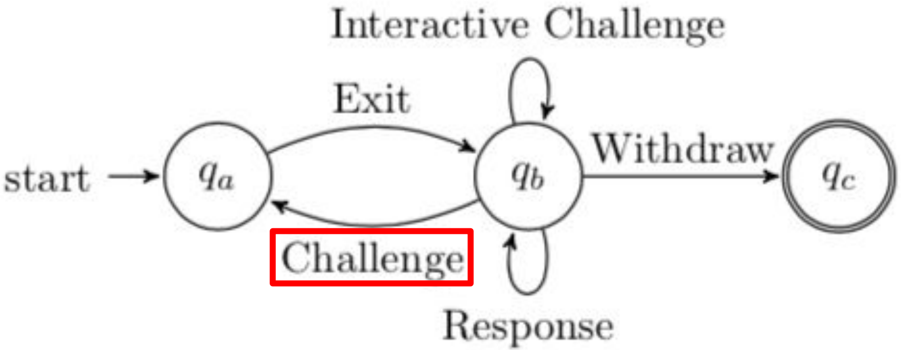


Exit = Spend from Deposit to Exit Script



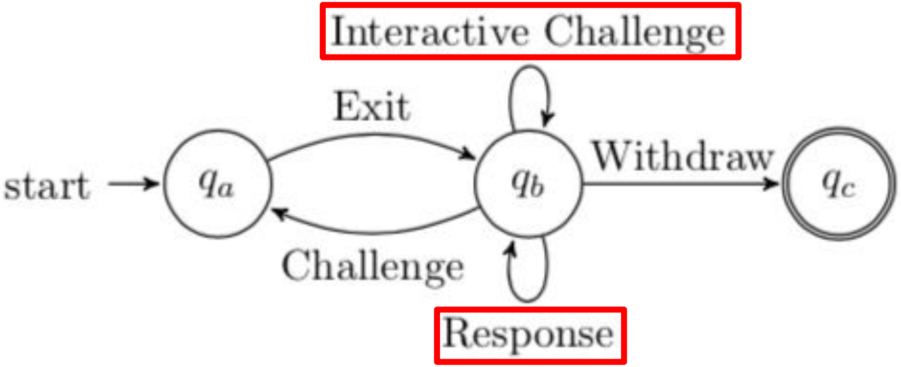
Spend to
`EXIT(parentIncludedTx, includedTx)`

Challenge Spent Coin / Double Spend = Spend back to Deposit



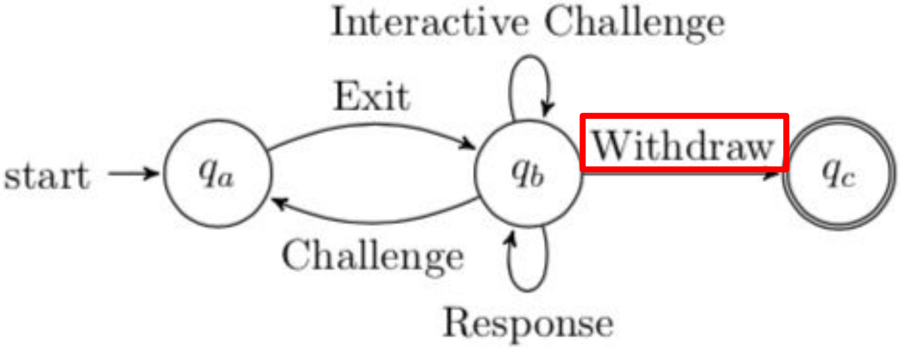
Spend to `DEPOSIT`, show `includedTx` according to exit game

Challenge Invalid History = Increment Counter, Response = Decrement Counter



Spend to `CHALLENGED′` , show `includedTx` according to exit game. New CHALLENGED state = previous state with 1 extra IF condition for the Response.

Witdhraw = Spend wherever after T if counter = 0



```
CSV 150 BENEFICIARY_ADDRESS CHECKSIG
```

Summary

- Off-chain fixed-denomination payments
- “Compression” mechanism (more txs settle per block)
- Operator can censor cannot steal (under liveness assumption)
- No on-chain transaction to join
- Can receive payments when offline
- Capital efficient
- Users must audit **Bitcoin-chain** for fraud (*light* client side validation)
- Implementation WIP! Complex & Secure scripts are hard. Has been implemented on Ethereum since June ‘18.
  Forked @kalewoof’s btcdeb for prototyping (https://github.com/gakonst/btcdeb, rust CLI interpreter for opcode experimentation WIP!)

Thank you for your attention

Q & A ?

@gakonst / me@gakonst.com

gakonst.com/scalingbitcoin2019.pdf

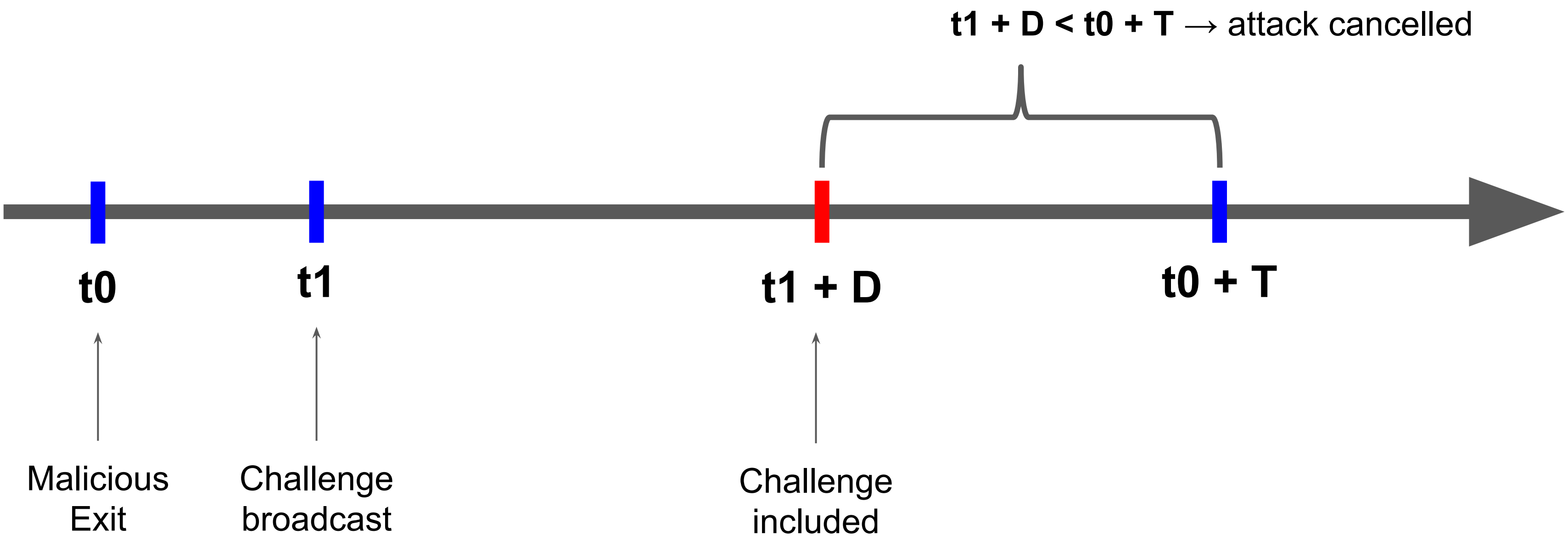
Appendix

Security & Incentive Compatibility
of Layer 2 games requirements*:

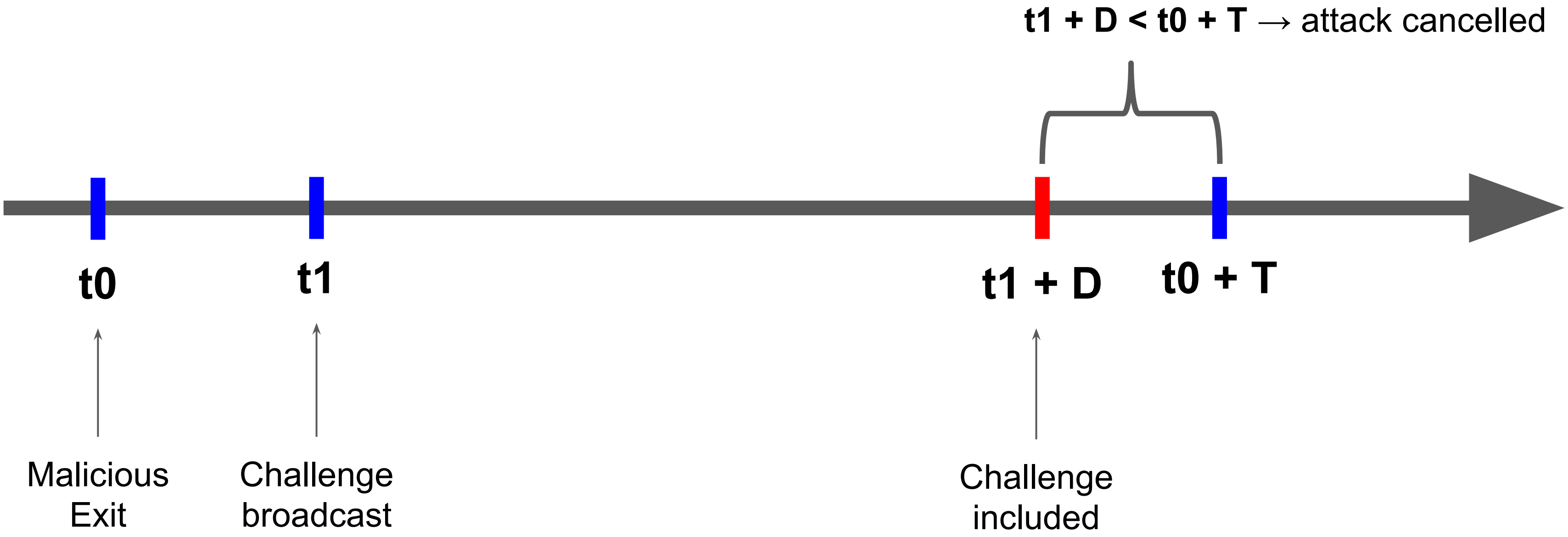
- **liveness (somebody must challenge)**
- **expected reward of attacker <=0**

*L2 games are implemented as deferred optimists:
https://medium.com/@decanus/optimistic-contracts-fb75efa7ca84

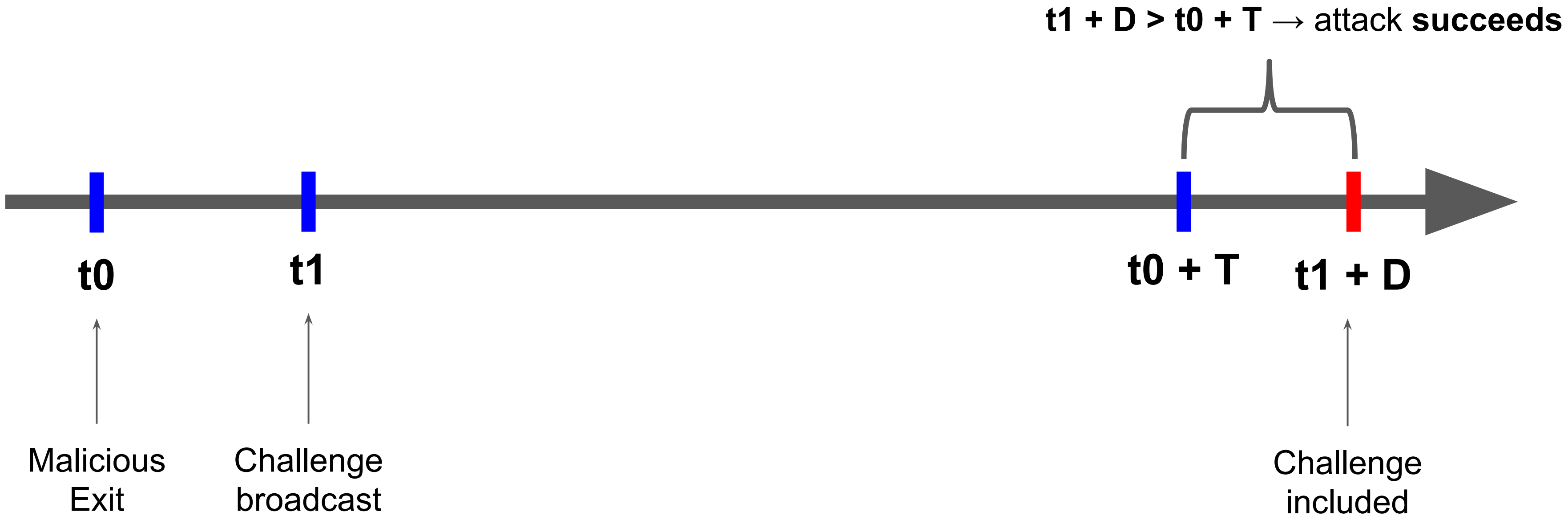
Secure iff challenge included before t0 + T



Secure iff challenge included before t0 + T



Insecure iff no challenge included before t0 + T



t1 + D > t0 + T → attack **succeeds**

t0

t1

t0 + T

t1 + D

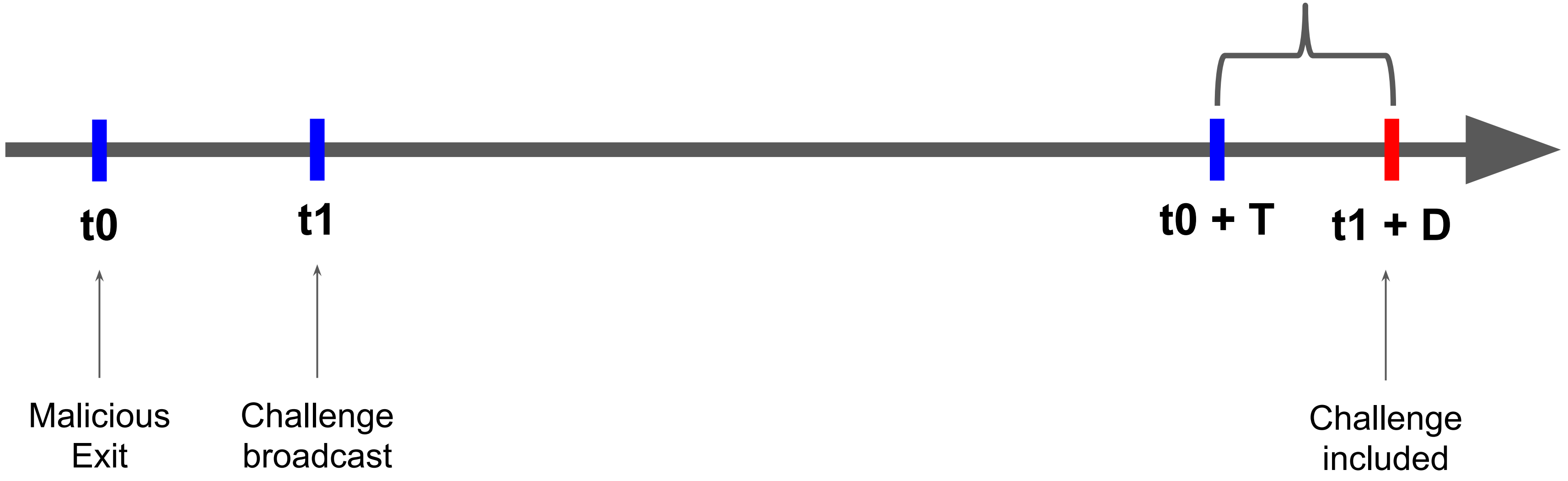
Malicious Exit

Challenge broadcast

Challenge included

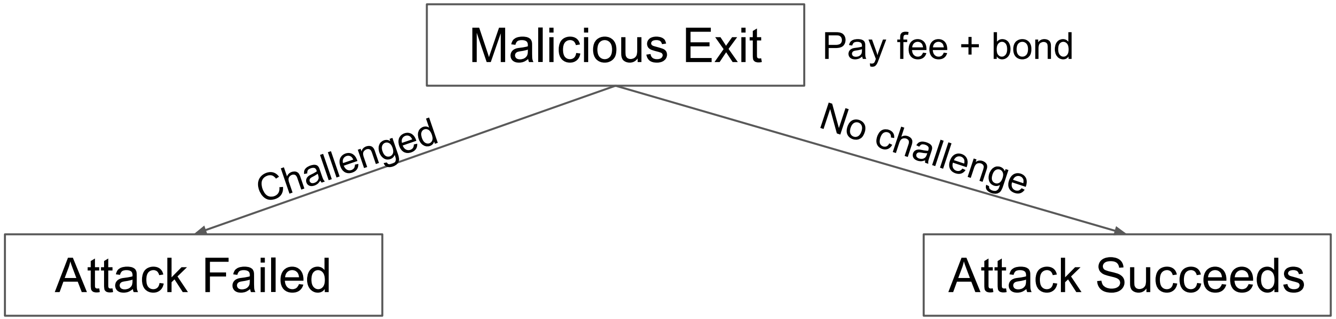
Insecure iff no challenge included before t0 + T

t1 + D > t0 + T → attack **succeeds**



Safety condition: D <= T + $\boxed{\mathbf{t0 - t1}}$ ← Liveness of observers

Attacker Decision Flow



Malicious Exit

Pay fee + bond

Challenged

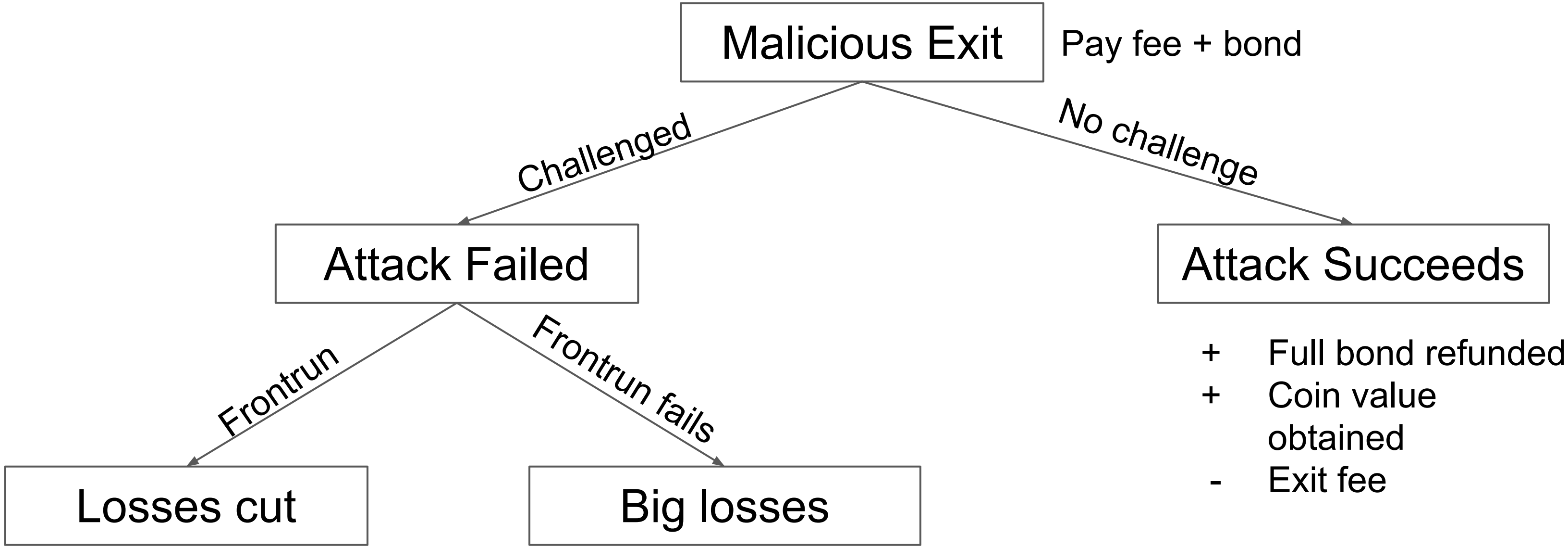
No challenge

Attack Failed

Attack Succeeds

- + Full bond refunded
- + Coin value obtained
- - Exit fee

Attacker Decision Flow



Malicious Exit

Pay fee + bond

Challenged

No challenge

Attack Failed

Attack Succeeds

Frontrun

Frontrun fails

Losses cut

Big losses

+ Full bond refunded
+ Coin value obtained
- Exit fee

- **a% of bond refunded**
- Exit fee
- Challenge fee

- **100% of bond lost**
- Exit fee
- Challenge fee

Incentive Compatibility of the Exit Game

$$E(R) = P(\overline{C})v \qquad\qquad \leq 0$$

No challenges = success:

- ↑ onchain congestion / censorship
- ↑ block withholding
- ↓ liveness of participants
- ↓ **challenge period *T***

Large T = Secure but bad UX!

Incentive Compatibility of the Exit Game

$$E(R) = P(\overline{C})v - \underbrace{[gas + P(C) * bond]}_{cost\ to\ attack} \qquad \leq 0$$

No challenges = success:

- ↑ onchain congestion / censorship
- ↑ block withholding
- ↓ liveness of participants
- ↓ **challenge period *T***

Large T = Secure but bad UX!

Cost to Attack =

- Tx fees (constant)
- **Fidelity Bond**
  (goes to challenger)

Incentive Compatibility of the Exit Game

$$E(R) = P(\overline{C})v - \underbrace{[gas + P(C) * bond]}_{cost\ to\ attack} + \underbrace{P(C)P(F \mid C) * bond}_{reward\ from\ frontrunning} \leq 0$$

No challenges = success:

- ↑ onchain congestion / censorship
- ↑ block withholding
- ↓ liveness of participants
- ↓ **challenge period *T***

Large T = Secure but bad UX!

Cost to Attack =

- Tx fees (constant)
- **Fidelity Bond** (goes to challenger)

$$P(F \mid \overline{C}) = 0$$

Attacker won't frontrun if nobody challenged

Frontrunning removes bond from cost if successful

Incentive Compatibility of the Exit Game

$$E(R) = P(\overline{C})v - \underbrace{[gas + P(C) * bond]}_{cost\ to\ attack} + \underbrace{\alpha P(C) P(F \mid C) * bond}_{reward\ from\ frontrunning} \leq 0$$

No challenges = success:

- ↑ onchain congestion / censorship
- ↑ block withholding
- ↓ liveness of participants
- ↓ **challenge period *T***

Large T = Secure but bad UX!

Cost to Attack =

- Tx fees (constant)
- **Fidelity Bond** (goes to challenger)

$$P(F \mid \overline{C}) = 0$$

Attacker won't frontrun if nobody challenged

Frontrunning removes bond from cost if successful

Burn part of the bond.

Plasma Cash → Fixed-denomination.
Arbitrary denomination payments?

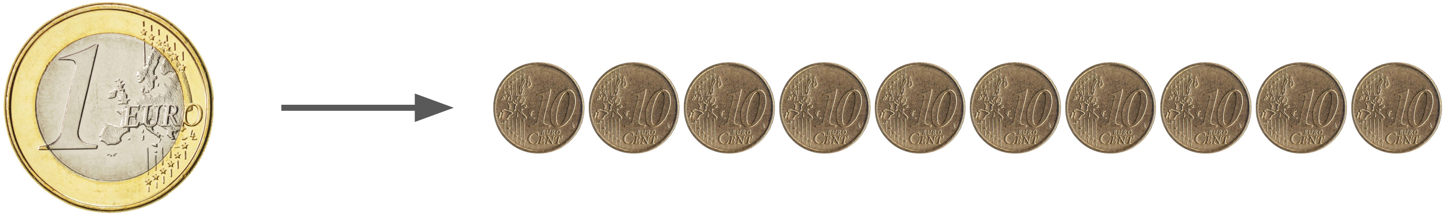
pg

https://github.com/plasma-group/plasma-core

Plasma Cash + Fragmentation = Plasma Cashflow

1 Euro

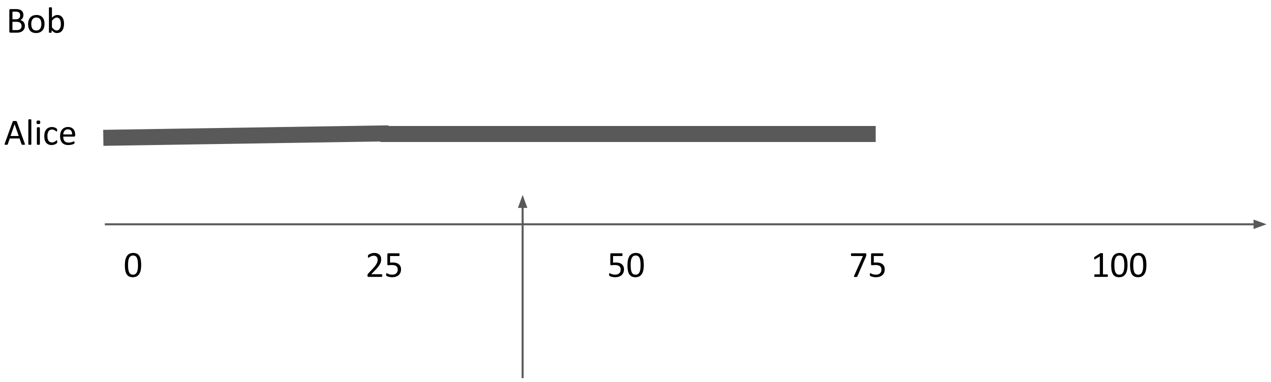
Plasma Cash + Fragmentation = Plasma Cashflow



1 Euro

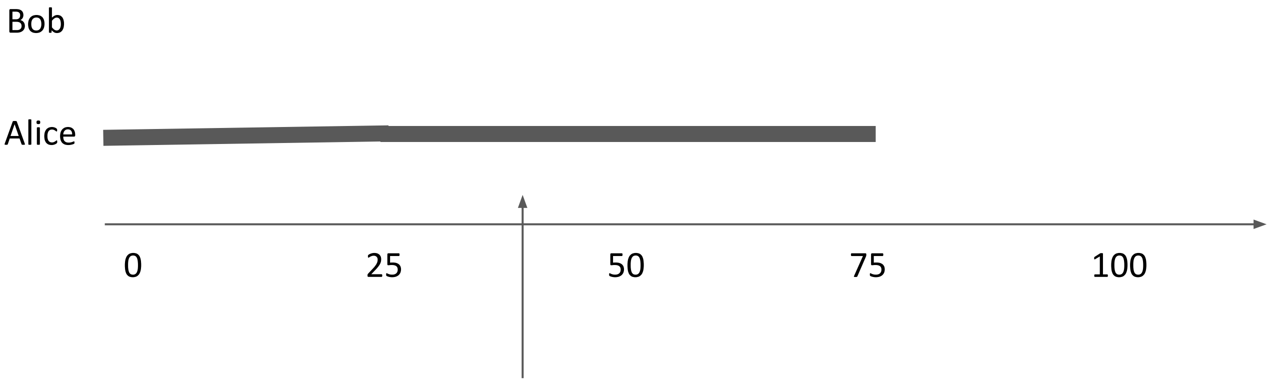
range of 10 x 10 cent **fragments**

A non-interrupted range can be transferred in 1 tx



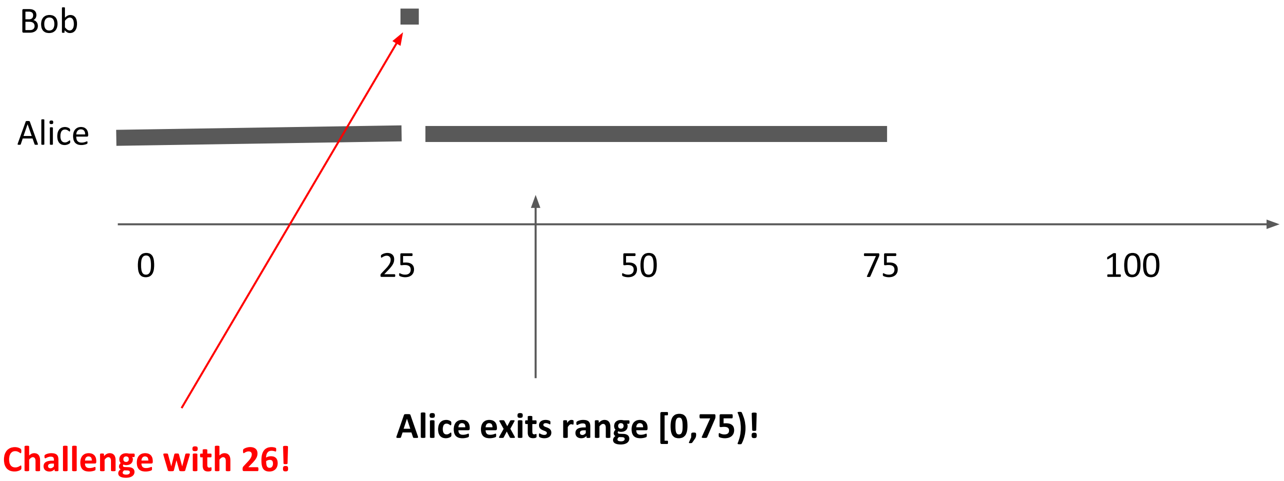
Alice transfers range [0,75) to Bob!

A non-interrupted range can be exited in 1 tx



Alice exits range [0,75)!

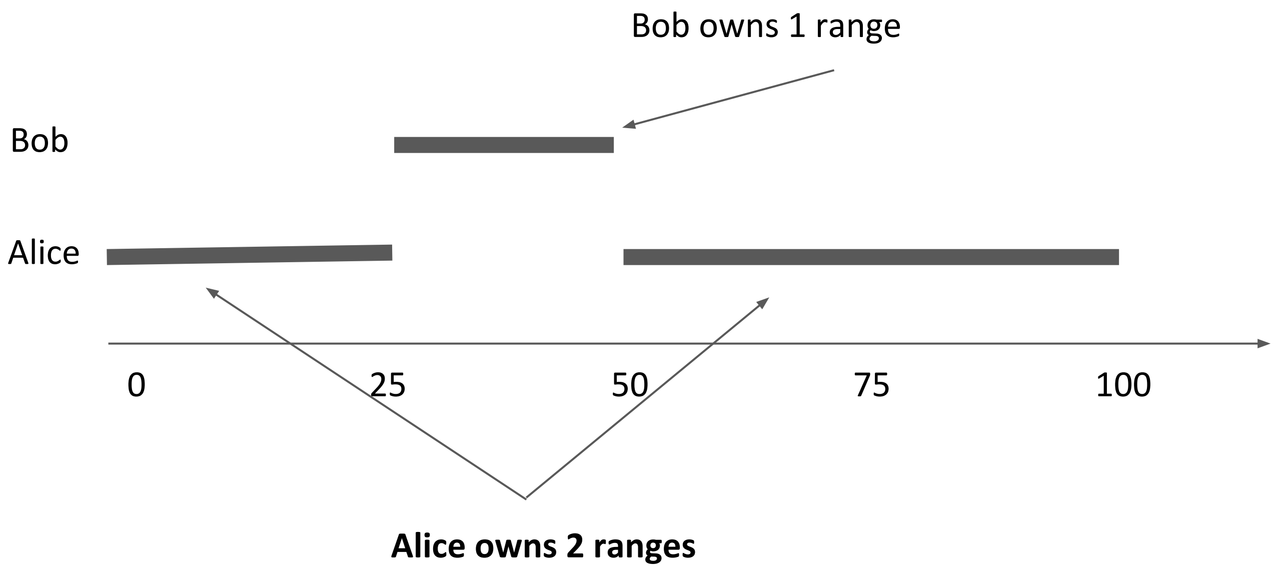
Any 1 coin inside the range is a valid challenge!



Alice exits range [0,75)!

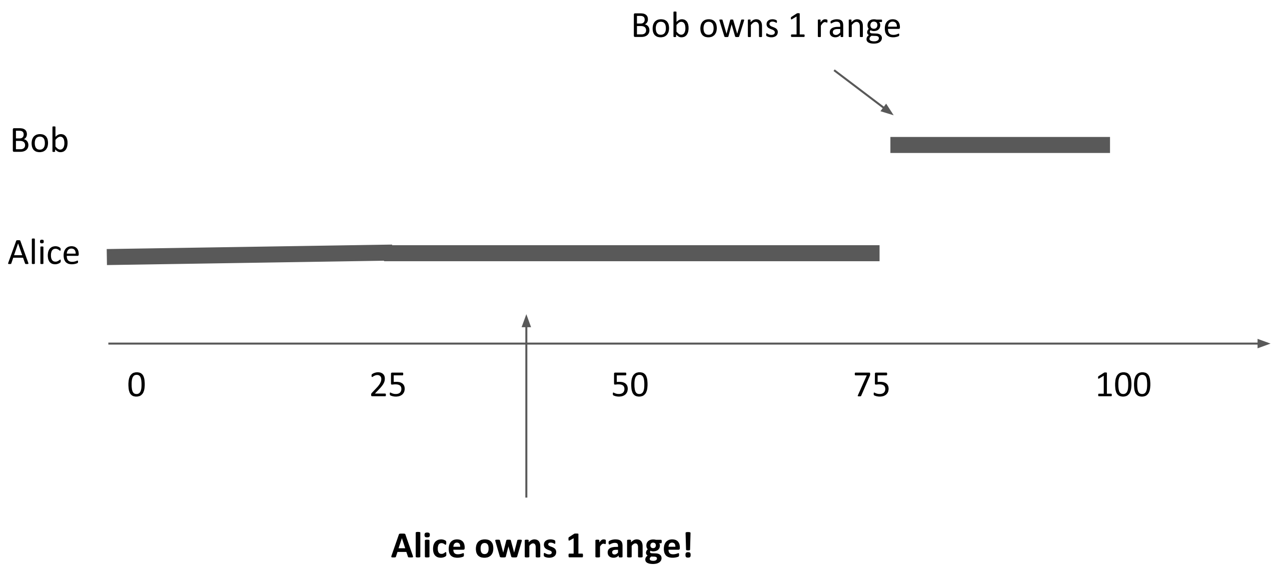
Challenge with 26!

Defragmentation of ranges



https://twitter.com/_sgtn/status/1100357379760091137

Defragmentation of ranges

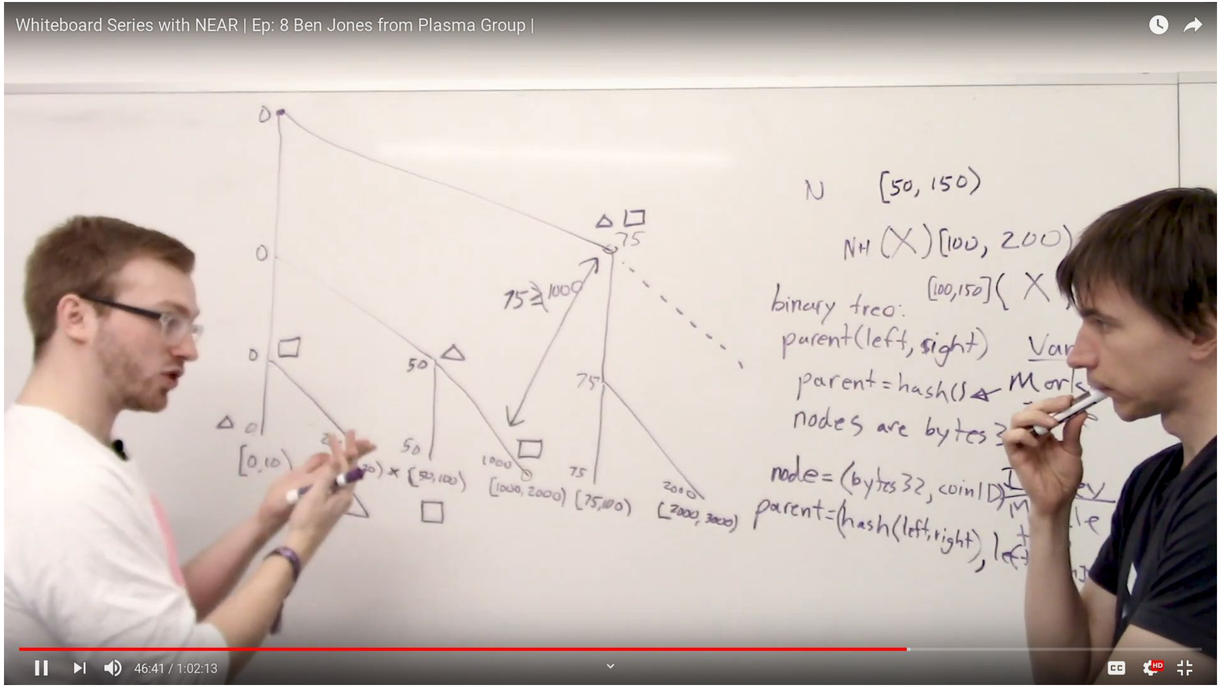


Alice owns 1 range!

https://ethresear.ch/t/plasma-cash-defragmentation/3410
https://ethresear.ch/t/plasma-cash-minimal-atomic-swap/3409

Merkle Index Tree

Inclusion / exclusion proofs **for ranges w/ light client support!**



https://www.youtube.com/watch?v=-8Jp7VjspQE