Multi-Party Channels in the UTXO Model

Challenges and Opportunities

Laolu Osuntokun

@roasbeef

Co-Founder & CTO, Lightning Labs

Table of Contents

I. **Single-Party Chans + The Promise of Multi-Party Chans (MPC)**

II. **UTXO Model vs Account Model for MP-Chans**

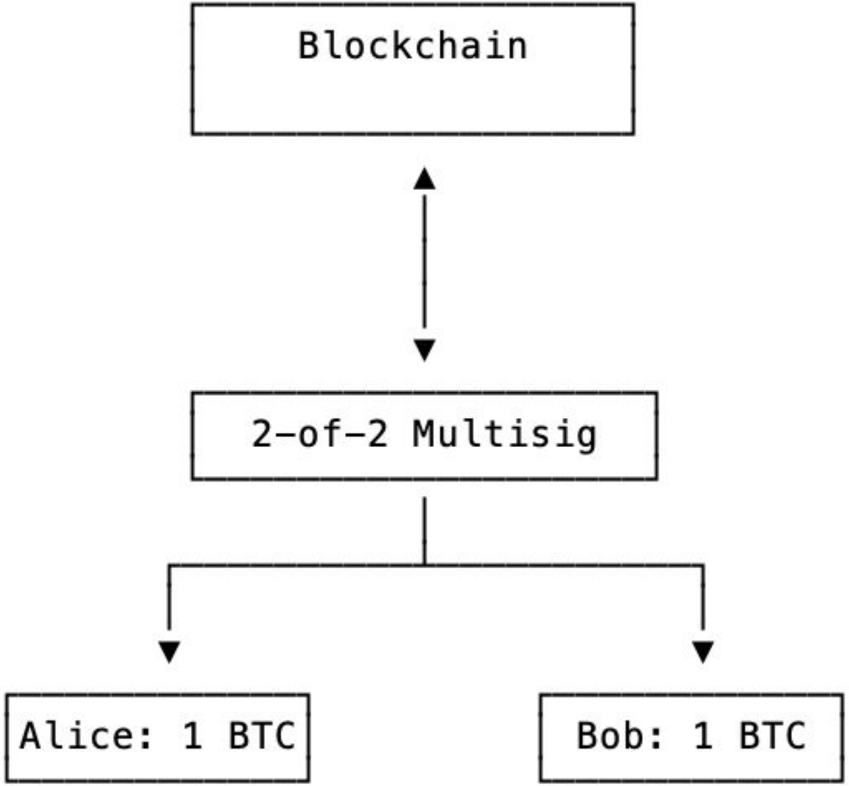
III. **Existing Constructions for UTXO-based MP-Chans**

IV. **New Directions in MP-Chans**

V. **Open Problems in MP-Chan Construction**

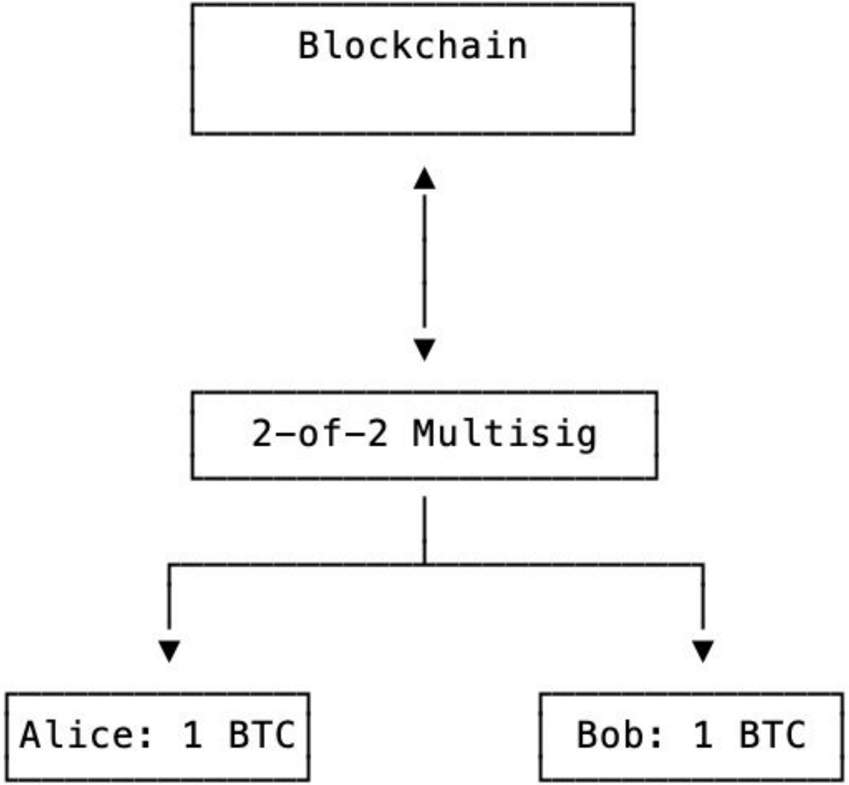
Single Party Chans - Overview

- **Emulate** a shared **account** using a 2-of-2 multi-sig
- On-chain **control transactions**:
  - Open
  - Cooperative close
  - Force close
  - Splice-In/Splice-Out
- **Rapid** off-chain balance **updates**
- **Atomic** conditional payments via HTLCs
  - Hash Time **Locked** Contracts
- **Bridging** channels via **HTLCs**
  - Starts to get more **network-y**



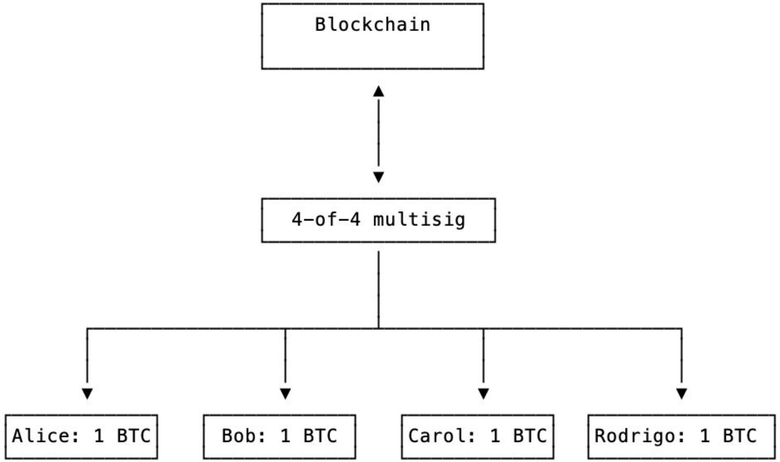
Single Party Chans - Limitations

- Flow of funds **constrained** by **topology** of channels
    - Requires ***planned*** **bootstrapping** effort for swift **onboarding** experience (autopilot)
- **Unable** to **dynamically create** new channels off-chain
    - Each **new user** on-boarded to LN **requires on-chain txn** (ignoring custodial wallets)
- Each channel **requires** a **single UTXO**
    - Can only be so many UTXOs in the system…



Multi-Party Channels - Opportunities

- **Generalization** of **two-party** contracts to **multi-party** contracts
  - Extends payment ability to allow **n-to-n interaction**
- No longer need a **new utxo** for **each channel**
  - **Single UTXO** potentially creates **1000s of channels**
  - multi-signature techniques, can make funding transactions appear as multi-input sweeps!
- Able to **collocate** into "**economic zones**"
  - Frequently transacting parties Likely save on networkwork level forwarding fees
  - **Off-chain** channel **creation/destruction**
- **Dynamic route creation** in the **Lightning** Network
  - Able to dynamically "tunnel" payments
- Applications:
  - **MMO** gaming Servers
  - P2P payment focused applications
  - Bill-splitting, etc



Multi-Party Channels - UTXOs vs Accounts

- Most **existing** constructions in the **account model**:
  - **Single contract** with "virtual" accounts within the contract
  - Existing constructions/deployments
    - **Plasma**
      - Hierarchical side chains with exit clauses, root chain stamped in main chain
    - **NOCUST**
      - Creates "bi-modal" accounts on-chain and off-chain
- **Challenges** in **UTXO** model
  - Lack of state in contracts seems to force **hierarchical constructions**
  - Hierarchical constructions can have **large on chain footprint**
  - **Limited scripting restricts** range of challenge proofs
- **Advantages** of **UTXO** model
  - Able to easily **create new contracts off-chain**
    - No need to "counterfactual instantiation" or w/e
  - Hierarchical states allow **flexibility** + **decoupled** updates

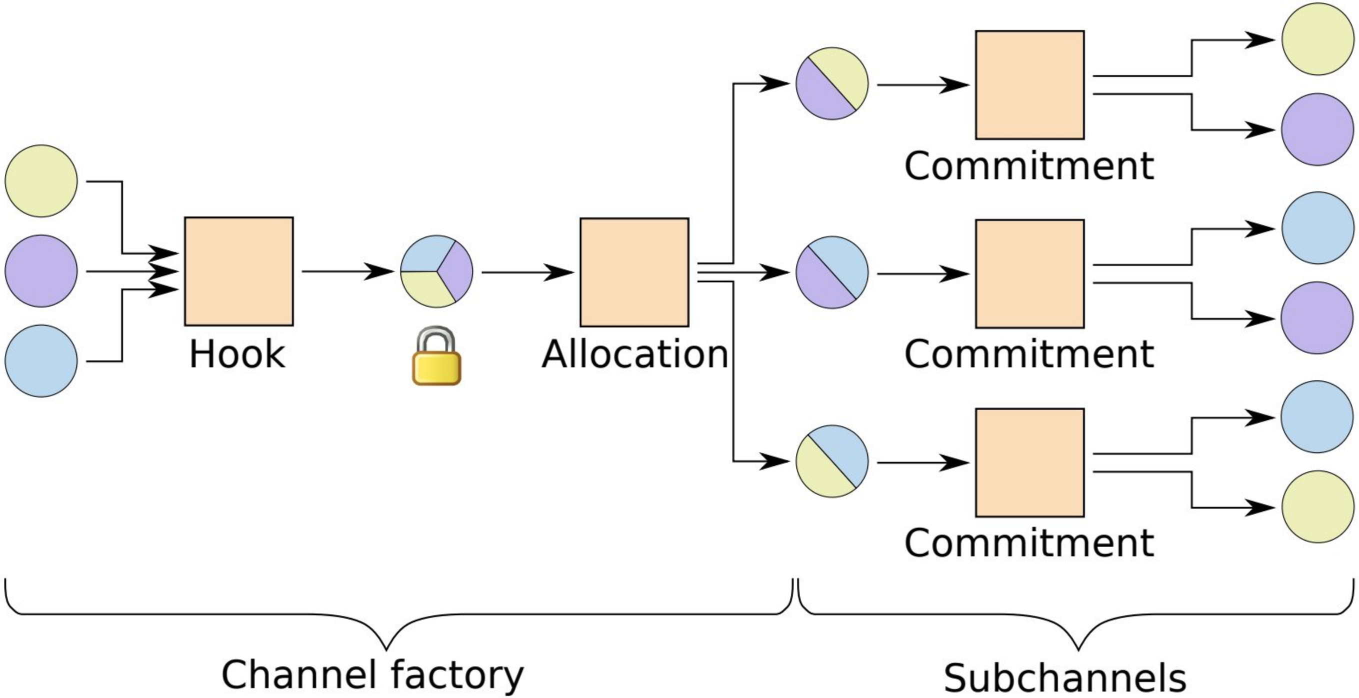
UTXO Based Multi-Party Channels - Lineage

- Duplex channels
  - Nested commitment replacement by relative lock-time
    - **Invalidation tree** recursively applies relative-lock time to achieve longer channel lifetime
  - Addition of kick-off transactions later allowed for indefinite channel lifetime
- Eltoo (or signed sequence locks!)
  - Commitment **replacement by version**
  - Addresses on-chain state blowup issue due to usage of invalidation trees
- Channel Factories
  - Framework for **hierarchical multi-party channels**
  - Originally used invalidation-trees for n-party commitments
  - Addition of eltoo reduces already large on-chain footprint in the worst case
- Lightning Factories
  - **Recently** published (like earlier this week)
  - Applies **replacement-by-revocation** to a channel factory-like framework
  - Utilizes **BLS signatures** to reduce communication complexity
  - Doesn’t appear to solve state blow up issues

UTXO Based Multi-Party Channels - Channel Factories

- Hierarchical **n-party** channel construction:
  - **Layers** of **intermediate** transactions creating various sizes of mult-sigs
  - Further down tree (towards leaves) # of keys in sigs grows smaller (**fan-out**)
- Channel Factory Terminology
  - Hook
    - Initial n-of-n multi-sig funding transaction
    - Requires **all parties** to sign-off for updates
    - Can utilize **key-aggregation/multi-signatures** to shrink to **single key**
  - Allocation:
    - **Sub-divides hook** into **smaller** multi-sig subset
    - Used to **shape structure** of relationships further down in tree
  - Commitment:
    - **Leaf nodes** of 2-party channels
    - Usage of eltoo at leaves allows for n+ leaf chans

UTXO Based Multi-Party Channels - Channel Factories



New Directions - New User Off-Chain Chan Creation

- Able to join new channels ***without*** on-chain transactions
    - Partially addresses **on-boarding problem** of new users to LN
        - “Alice has no Bitcoin, how do we get her onto Lightning **without** an on-chain transaction”
    - Simply **modify** existing **allocation** to **add key of new user**
    - User then able to update channel in place, never touching chain!
    - Allows for **dynamic growth** of # **of users** in channel, UTXO growth contained!
- Requires **new trust assumption**
    - Able to obtain valid **channel audit proofs** from **threshold** of active **users in channel**
    - Need to ensure being “teleported” into **latest valid state** within channel
    - As all updates off-chain **can’t use raw chain to verify** “freshness” of proposed state
        - MP-Chans like icebergs, **can only see hook**, not below to allocation/commitments
        - Similar to “weak subjectivity” assumption in PoS
- Can also **splice in/out** new funds/participants via **sighash no_input**

New Directions - Threshold Channel Audit Proofs

- Intra/inter multi-party channel operations, require **“freshness”** arguments of channel state
  - Otherwise can sign away output or state to/from a channel that **actually doesn’t exist!**
  - Typically only have **limited visibility** into surrounding channel tree
- **Audit** proof:
  - Introduce new **modified sighash**: single sha instead of double-sha
  - Require entities from leaf to root/hook to sign description of channel state
    - Need enough information to be able to reconstruct txid of txns
  - Proof verifier specifies **threshold** of parties at each internal branch (n-of-n multi-sig)
- Required for:
  - New user **off-chain channel creation**
  - Cross **sub-tree swap** operations

Lightning Cross Over - Route Tunneling

- The current LN graph is generally relatively **static**
  - Channels take up to **6 confs** before becoming routable by remote parties
  - Channel closes can take **10 of minutes** to execute
  - Graph **verified** by nodes to prevent DoS/sybil attacks
- Multi-party channels allow for **dynamic channel creation**, there for **dynamic route creation!**
  - Channel relationships in mp-chans exist in “another **dimension**”
  - Can be used by nodes “above ground” to **advertise short cuts** route that **tunnel** through channel formation
  - Able to create **new channels** in seconds to satisfy directional flow above above ground
- Requires **distance-vector** like **announcements**
  - In contrast to **circuit-switching** widely utilized today
  - Supplemented by proposals for **balanced congestion aware** packet switching within the network
- Can also be used as a **bridge** to multiple mp-chans
  - Used recursively to **dramatically reduce** network **diameter**

Lightning Cross Over - Multi-Party Nodes

- Alternatively, can **advertise** mp-chan **as single regular channel**
  - **Series** of **smaller** mp-chans **linking** either single chans or other mp-chans
  - **Channel “colony”** addressed externally by **single node public key**
- Allows multiple nodes to **aggregate channels** and **combine liquidity**
  - **Shrinks** the size of the **public graph**, 100s of channels seen as a single channel
- Current protocol implements **limit** on **# of outstanding HTLCs** per channel
  - Usage of **AMP** combined with a **max HTLC size** (essentially an **MTU**) results in **constrained commitment space** network-wide
    - Limits set for **single transaction penalty** (**966** HTLCs) can easily be raised to target **max transaction weight policy l**imit
  - Mp-chans essentially allow queue size to grow dynamically via **nested commitments**!
    - Similar trick (**indirect commitments**) can be used for regular channels as well

Lightning Cross Over - Hierarchical Prefix Addressing

- How to handle **receives** over **multi-node** (network aggregated) mp-chan?
  - Today HTLCs **targeted** at **single** destination **public key**
  - **Multi-node channels** potentially contain **hundreds of nodes**
- Solution:
  - Individual parties within the mp-chan **self-organize to assign address** based on up-to-date **structure of the commitment tree**
  - **Destination** address within commitment tree **placed in EOB** (extra onion blob)
  - Parsed from **left-to-right** respecting **fan out of intermediate** allocations to dispatch payment to proper leaf node:
    - Ordering of **keys** in allocation **sorted** to allow deterministic parsing
    - Example for 8 -> 4 -> 2 (x4) channel:
      - [10][1]

Cross Channel Swaps via Swaptions

- Possible to exchange positions within a particular channel, or even **trade positions** within distinct channels
  - Swap itself creates **new channel state**, no need to thread prior history
- Vanilla atomic swaps have free option issues as **single party** can **halt execution**
- **Atomic Swaption**:
  - Alice **sells** Bob the **option** to **swap positions** within same/distinct channel
  - Regular atomic swaps use a **single secret**
  - Atomic swaptions instead involve **two distinct secrets**
  - **Two layers** of transactions:
    - **Acceptance** layer:
      - Alice can **accept** by **revealing** secret **A** which leads to second-layer that **unilaterally** pays Bob the **premium**
    - **Exercise** layer:
      - Bob can **exercise** the option **till expiration** by **revealing** his secret **B**
- Potentially allows the **sale/transfer of channels** within distinct channels!

Channel Orchestration Servers

- **Distributed** version requires **quadratic communication** for **re-allocations** scaling with number of participants in internal node
  - Shifting to **single-key n-of-n** (schnorr) requires **additional round trips** for each signature
- Can use a **message passing server** to reduce to **linear communication** between parties
  - Channel participants use **server as rendezvous** location over Tor onion services
  - **Leaks timing** information of updates, but server doesn't necessarily know which channels are being updated
    - Participants can send/receive **dummy** messages **mix-net** style

Channel Orchestration Servers - Offline Payment Receipt

- Why not also use orchestration server as **offline mailbox**?
  - Participants **pay orchestrato**r to **deliver** message with set deadline
  - Allows for **quasi-offline paymen**t sending/receipt
  - During **clearing phase** (HTLC add), if participants not offline within threshold, **cancel** back
  - During s**ettle phase,** fully async as **receiver** only comes **online** to **reveal** secret
- Similar model **possible** over **"regular" network**, but would need to **pre-pay** to **several parties** to compensate for **longer** HTLC **lifetime**

Open Problems

- **Cut-thru** to **reduce** on-chain **footprint** in **mass exit** case?
- Usage of **covenants** to allow hook transaction modifications **w/o all parties involved**?
- **Health checking** protocol to splice out inactive parties within allocations
- **Language** for **expressing** complex multi-step re-allocations and swaps?
    - BitML?
- Efficient execution of **fees+timelocks** in **packet-switched** model?

Thank You!

Questions?