Seed Based Payment Channel Recovery

David Vorick

Background

Lightning network is a payment channel network that improves Bitcoin's transaction scalability

Scalability comes from continually updating these channels off-chain

Problem

Keeping these payment channel midstates is critical for the user's security.

Loss of the mid-state can result in an inability to use the channel, and can allow your counterparty to steal funds by posting an old state.

User backups need to remain up to date.

Partial Solution: Watchtowers

Watchtowers track midstate for the user.

Watchtowers can penalize counterparties that cheat.

Watchtowers cannot post the final state of a transaction if both the user and the counterparty are offline.

Watchtower Complications

Watchtowers need to store data for every mid-state of a channel, which is expensive for high usage channels.

Watchtowers need to have high reliability and uptime to be highly effective.

Watchtowers are non-trivial infrastructure to add into the lightning ecosystem. Practical, but non-trivial.

Alternate Solution: Seed-Based Midstate Recovery

Our goal is to provide the user with a method to recover their channel mid-states that does not depend on any external party such as a watchtower.

Major Assumption

This technique depends on a user forming channels with a hub which can be expected to be more reliable. In a purely peer-to-peer environment, the “hub” can be substituted with a watchtower that gets updated after every payment channel transaction.

Future research may enable this research in a pure peer-to-peer environment without a watchtower.

General Strategy

The user will leverage their payment hubs as storage devices for all recent payment channel information.

Each time the user connects to the hub, the user will require the hub to provide all information.

All interactions are identical, even after the user has lost data. The hub does not know when there is an opportunity to cheat.

Information Stored

The hub will be storing the most recent midstate of the channel.

The user can also require the hub to store the most recent midstate of other channels (other hubs, and other p2p peers).

The data will be presented as a single encrypted blob of fixed size, suggested 40 KiB.

Information Stored

The blob can be encrypted using a password derived from the user's wallet seed and the hub's public key.

40 KiB leaves room for the midstates of a large number of channels, without being burdensome to the hub.

The hub can support hundreds of thousands of users with just a few GiBs.

Information Stored

The blob will contain a counter/version, allowing the user to determine which blobs are most recent.

Because the blob is fixed size, there is also potentially room in the blob for additional arbitrary data.

User Requirement

The user will store the information that is kept in the encrypted blob. This allows the user to verify that the hub is honest in the typical case.

Payment Protocol (Normal Operation)

The payment protocol will require that the hub provide the user with the most recent version of the 40 KiB blob before the user makes any transactions.

If the user has not experienced data loss, the user will decrypt the blob and compare it to the blob that the host is supposed to be storing.

Payment Protocol (Normal Operation)

If the user has channels to other hubs, the user will open a connection with them and request the most recent blob from each of them as well.

If the blob matches, the user will prepare the desired transaction and give each honest hub a new blob to store which contains the updated midstate.

Payment Protocol (Normal Operation)

After all hubs have confirmed receipt of the most recent blob, the finalized transaction can be sent. This completes the protocol.

Payment Protocol (Dishonest Hub)

If any hub is dishonest and provides an outdated blob, the user completes a transaction and later requests that the hub close the channel.

When the channel is closed with the outdated state, the user can submit a penalty transaction.

Protocol Extension

The protocol can be extended to have all hubs sign each message with a timestamp. If a hub ever attempts to act maliciously, the user has explicit proof that can be provided to other users on the lightning network.

Payment Protocol (User Recovery)

If the user has lost data, the user can still perform the payment protocol.

The payment protocol starts like normal. The user pretends that all data is available, and requires the hub to send the most recent version of the blob.

Payment Protocol (User Recovery)

After receiving the blob from the hub, the user will request the same blob from all other hubs that the user has channels with.

So far, there has been no deviation from the typical protocol. The hubs have no way of knowing yet that the user has lost data.

Payment Protocol (User Recovery)

The user has now received encrypted blobs from all hubs.

The user is able to decrypt the blobs with just the wallet seed, and then cross-reference the counters and payment channel states in each blob.

So long as at least one hub is honest, the user is able to perform a successful payment channel midstate recovery.

Peer to Peer Channels

So long as the user has a channel with at least one hub, the user can use this protocol to store payment channel midstate updates for all peer to peer channels.

Tradeoffs Overview

The biggest tradeoff is that all hubs must be contacted each time a payment channel update occurs.

A small tradeoff is that each hub must now store 40 KiB per user. This should be trivial, especially if users are paying transaction fees to the hubs.

Incentives Observation

An honest hub can provide the user with information that potentially allows the user to penalize a competing hub.

This acts as a mild incentive for hubs to be honest.

Security Limitation

If all of a user's hubs are collaborating dishonestly during a user recovery, the user will be unable to detect the malice and becomes vulnerable to penalty transactions from the hubs.

Seed Based Midstate Recovery: Missing Pieces

This presentation has covered how to recover a midstate from a hub.

If the user has lost all data and has only a wallet seed, the user needs to undertake the task of figuring out which on-chain payment channels belong to the user, and which hubs are the counterparties to each channel.

Payment Channel Coloring

The user can color each opening payment channel transaction with some extra identifier data in the transaction.

The identifier can be a message encrypted using a password derived from the user's seed and the output id of the first output in the transaction.

The identifier will be called a "color".

Payment Channel Coloring

Every transaction will have a different color. External parties will be unable to link related transactions based on the color.

The owner of a transaction can identify all of their payment channels using nothing but the public blockchain and their wallet seed.

Counterparty Discovery

The on-chain transaction will need to contain a reference to a public key of a counterparty.

If privacy is desired, this reference can be encrypted using the user’s wallet seed.

Counterparty Discovery

The public key can be used to look up the counterparty in a public key infrastructure.

This PKI could be stored on-chain, on a side-chain, or could be a more traditional PKI.

The PKI must map from the public key to an IP address or hostname that can be used to contact the counterparty.

Tradeoffs

The transaction coloring and on-chain public key each increase the on-chain size of a payment channel.

These strategies can likely be deployed in ways that do not impact privacy.

Better than Watchtowers?

Requires hubs.

Requires slightly increased on-chain storage.

Transacting requires contacting more parties.

Probably simpler than watchtowers in practice.

Not a perfect replacement for watchtowers.

Future Research

The increased on-chain transaction sizes can probably be reduced.

The ‘contact all hubs each transaction’ requirement can probably be reduced.

Better defined fee mechanisms for hubs.

Elimination of hubs as a requirement.

Questions?