Below are some musings that arose from discussions with @spalladino on how the current note discovery scheme could be made more robust in terms of tag index coordination from the basic model we have today.

Note Discovery and Indices Today

The current note discovery scheme relies on tags being computed deterministically by both sender and recipient after agreeing on a shared secret by hashing the secret with an index

counter. Both sender and recipient have to keep their counters in sync for note discovery to work: if the sender emits logs with indices that are too high, the recipient may not find those as it will only be looking for tags associated with smaller indices. Additionally, the sender must not repeat indices as that will cause a privacy leak by linking two sets of notes and transactions.

The initial version of note discovery will be relatively simple: the sender will only ever increase its indices and keep track of the last one it used, and the recipient will search for tags with increasing indices and keep track of the largest one it's seen.

This simple mechanism easily results in problematic situations due to no fault of either party: a sender may send two transactions and have the first one revert, creating a gap in the received indices, or these two transactions may be mined out of order, creating both a temporary gap and a perceived decreasing index.

The current plan is to have the recipient be lax, and search not just for the next expected index but also for indicest past that (to account for gaps) and below that (to account for perceived decreasing indices). The sender will check against the node for the largest index and use that if it's larger than its copy (to be able to migrate accounts to other devices), and the recipient will reset all indices if it ever detects a chain reorg.

A Path Forward

What's been described above suffices for early releases, but is not robust enough for production usage. There's big risk of a sender using indices too low or high, resulting in undiscoverable notes. Low indices could potentially be recovered if the recipient forced an index reset, but high ones can only really be fixed by sending more notes or increasing the recipient search window size. This led us to work on a better scheme.

We propose that the app circuits notify the PXE whenever an index is added. PXE will then store the associated transaction alongside the used index in its database, and then produce an index status based on the status of the transaction. A potential list of statuses might be:

1. free: unused for any tx
2. sent: the tx has been broadcast but not included in a block
3. mined/proven: the tx has been included in an L2 block that is not yet L1-finalized
4. finalized: the tx block epoch has been L1-finalized
5. dropped: the tx was sent but never mined, and the tx max-block-number has expired.

Statuses typically progress to finalized as time goes by (assuming both chains are finalizing). Reorgs can cause txs go to from e.g. mined to sent. Because this happens on chain, both the sender and recipient will eventually agree on the status of these transactions (the recipient will find them as they are the txs that include tagged logs).

The main idea once we have this is the following: the sender will refrain from using more indices if the index to claim is more than some amount N of slots away from the first non-finalized index. In order words, there can only be N in-flight tags at any point in time. This might seem restrictive, but we need to consider that tags are per (sender, recipient, app) tuple, and transactions get finalized after ~50 minutes in the worst case.

The one instance in which multiple tags might be required is when interacting with oneself (since e.g. token transfers to any

recipient trigger change notes), but since in that case we're both sender and recipient we don't really need that much help coordinating, and we could special-case it.

Because the recipient knows about N, it knows that it never needs to look for more than N tags after the first unfinalized index, and given sufficient time (for txs to finalize) all notes will be found without any on-chain action from neither sender nor recipient

.

Known issues

Mined transactions might revert, but we still should not use these indices. Even if the reverted tx is finalized, the tag is still publicly available as part of the TxEffects of the private component of the reverted transaction, which means that reusing it

would link the two transactions. Perhaps the node should also track these, so that the recipient can know to skip them.

Expired sent transactions that were never mined are more problematic: they will never go on-chain (since they're invalid), but some

people may have seen them and therefore their tags, given we did broadcast them. Do we reuse the indices and risk linking? The alternative is to somehow communicate to the recipient that they should ignore some index, but this needs to be done over an empty transaction, so that it is not linked to anything else.

Finally, this doesn't really help when sending transactions at the same time from multiple devices. I don't think there's much we can do about that really - we're doing a best-effort approach without any out of band communication by checking for mined transactions in the node (same as Ethereum wallets when choosing a nonce), and anything beyond that seems too complicated.

Disclaimer