Ledger Loops: Debt loops across local ledgers (Work in Progress)

Michiel B. de Jong

November 2016

Abstract

LedgerLoops is a protocol for clearing debts and credits across multiple ledgers. Unlike Bitcoin, which introduces one global public ledger, LedgerLoops acts on a loop of two or more local ledgers. This whitepaper introduces the concept of cryptographically triggered debts, which is at the core of LedgerLoops, and explains how they can be represented in a LedgerLoops contract. It also defines a messaging protocol which can be used to send these contracts along the ledger loop in a peer-to-peer fashion, and a decentralized algorithm to find cycles in a debt graph.

1 Cryptographically Triggered Debts

A debt can be represented by an "I owe you asset A"-message ("IOU" for short), sent from the debtor to the creditor. A cryptographically triggered debt would take the form "I owe you asset A from the moment person X signs a certain cleartext". It doesn't matter who this person X is (they might be a complete stranger). In fact, person X is only identified by a public key. The cleartext holds no meaning either, and can be chosen at random; it only acts as a trigger to activate the debt.

A cryptographically triggered debt does not immediately give the receiver a claim to the asset that would be owed by the sender; however, if the receiver can present cryptographic proof that at any point in time, person X signed the cleartext in question, this triggers the sender's debt into action. So a cryptographically triggered debt, combined with a valid signature for the cleartext it refers to, created with the key pair it refers to, is equivalent to a standard IOU. We say the existence of a valid signature *activates* all all cryptographically triggered debts that refer to this cleartext and this public key.

Settling a cryptographically triggered debt (after it has been activated) works the same way as settling a standard IOU: the IOU-receiver sends a

message back to the IOU-sender, stating that the IOU-sender no longer owes the assets mentioned in that specific debt statement.

2 The LedgerLoops Contract Format (version 0.5)

A LedgerLoops Contract is used to define cryptographically triggered debts. It refers to exactly one combination of a `cleartext` and a `pubkey`. It contains the following data (in UTF-8 JSON format):

```
{
  "protocolVersion": "ledgerloops-0.5",
  "msgType": "cryptographically-triggered-contract",
  "keyAlgorithm": "ed25519", // TODO: add requirements about key
      format, hashing, padding, etc.
  "pubkey": <some public key, compatible with keyAlgorithm>,
  "cleartext": "Go!",
  "debtor": <a string which both parties understand to identify the
      debtor>,
  "beneficiary": <a string which both parties understand to
      identify the beneficiary>,
  "deliverable": <a string which both parties understand to define
      both the maximum and minimum conditions for considering this
      debt paid>
}
```

3 The Algorithm

To get a feel for how the LedgerLoops algorithm works, consider the following minimal example, involving three ledgers:

- Bob and Otto maintain a peer-to-peer ledger, on which Bob owes Otto a banana.
- Otto and Michael maintain a peer-to-peer ledger, on which Otto owes Michael an orange.
- Michael and Bob maintain a peer-to-peer ledger, on which Michael owes Bob a mango.

Bob, Otto, and Michael can each only see two of these three ledgers, so none of them know that a debt cycle Bob -¿ Otto -¿ Michael -¿ back to Bob exists. To find this out and settle all three debts, they send the following messages:

3.1 Search round (debtor to creditor)

Bob, Otto and Michael all send the following message to Otto, Michael and Bob, respectively: "I have you as a creditor, but I also have at least one debtor, who may have other debtors, etc., which may eventually lead back to you."

In the actual implementation of the search round, a second message type is used - one like the one above, indicating "Yes, I'm interested in moving on to the probe round", and one indicating "No, count me out". This round only establishes a best-effort statement of intent between peers, and its only goal is to reduce the search space for the probe round.

3.2 Probe round (creditor to debtor)

Bob, Michael and Otto all send the following message to Michael, Otto and Bob, respectively: "Right, so you said you have at least one debtor; try sending them this token, instructing them to forward it, so I can see if it comes back to me." (accompanied by a long random string which is generated by Bob).

In the actual implementation, two tokens are used, one identifying the Depth-First-Search tree, the other identifying (backtrack-)paths along this tree. After this round, if the token string used was long enough to be unguessable, and the token makes it back to Bob, then Bob knows that a potential ledger loop exists (even though some participants may still pull out of the negotiations in a later phase).

3.3 Negotiation Round (creditor to debtor)

Bob now generates an asymmetric cryptographic key pair, keeps the private key safe, and sends Michael a LedgerLoops contract mentioning:

- the public key from the keypair,
- himself as the debtor,
- Michael as the beneficiary,
- as the deliverable he describes cancelling out Michael's mango debt.

Note that Bob does not intend to give Michael the equivalent of a mango, he instead states that he's willing to cancel Michael's debt if this cryptographically triggered debt gets activated. So maybe 'cryptographically activated debt cancellation' is a better name in this example than 'cryptographically triggered debt'. In general though, any transaction that updates the private peer-to-peer ledger between Bob and Michael could be sent in this way, as long as the recipient Michael is also the person benefitting from the ledger update (the beneficiary).

Michael then sends an analoguous LedgerLoops contract to Otto (replacing the names and the asset referred to), and Otto does the same again, so that Bob now has a Cryptographically activated debt cancellation from Otto, and holds the private key that activates it.

This round is called the negotiation round because, in future versions, peers may get a chance to haggle over exchange rates. That way, agents may for instance charge small transaction fees for converting between currencies - the smaller the transaction fee, the higher the chance all peers in the ledger loop accept the offer. But for now, a transaction fee of zero works best. :)

3.4 Settlement Round (debtor to creditor)

In this round, the messages travel along the ledger loop in the opposite direction again: from debtor to creditor. First, Bob signs the cleartext, and presents Otto with the signature, thereby triggering Otto's contract and claiming his deliverable of cancelling out Bob's banana debt.

Now that Otto has received the valid signature from Bob (even though he doesn't know the public key and signature belong directly to Bob), he can use that to tell Michael to cancel out to Otto's orange debt.

Michael is also happy, because now that he received the valid signature through Otto, he can tell Bob to cancel the mango debt.

All three debts have now been settled, and the goal has been achieved.

In this case the ledger length was quite short, but none of the participants know this precisely. Participants can roughly tell that a loop where the rounds take a long time is more likely to have more participants than one on which message loop around quickly, but this information is obfuscated by differences in speed of the communication network, and by participants who choose to wait a little bit before forwarding a messages.

No central authority or Unique Node List was used, each participant only ever trusted their own direct peers, and none of the participants exposed the contents of their private ledgers to the rest of the network.

Also, nobody except Bob knows who the public key belonged to that triggered all three LedgerLoops contracts, and each peer only exposed their identity to their direct neighbors in the ledger loop. Unless participants volunteer to make this public, nobody (except for each participant's direct neighbors) knows exactly who particpated in which ledger loop.

4 Conclusion

This is a work in progress. Full implementation details of LedgerLoops 0.5, as well as an initial discussion of security considerations, will be documented soon, once the list of unresolved issues stabilizes a little bit more: https://github.com/michielbdejong/ledgerloops/issues