A trusted latency monitor service, for preventing abuse in a Lightning-based peer-to-peer exchange

C. J. Plooy (Bitonic)

11 may 2018

Contents

1 Introduction

1.1 The Lightning network

The Lightning network[1] forms a layer on top of a crypto-currency, where participants create *microtransaction channels*, which together form a network that allows payments to be routed, over multiple hops if necessary, from one participant to another. On a route A-B-C-D, A and B share a channel, B and C share a channel and C and D share a channel; a transaction from A to D is performed by updating the channel balances such that B receives funds from A, C receives funds from B and D receives funds from C. The amounts of funds transferred is equal, except (optionally) for a small fee charged by the intermediaries B and C; the net result (except for the fees) is that D receives the funds sent by A.

An important feature of the Lightning network is that the payer and the payee do not have to trust the intermediate nodes: transactions are set up in such a way that intermediate nodes can not receive incoming funds without also sending out outgoing funds. This is accomplished with HTLCs (Hash Time Locked Contracts): smart contracts (contracts written in the scripting language of the underlying crypto-currency) which specify that one party can access the locked-in funds, but only if the pre-image of a certain hash is provided, and the other party can access the funds, but only after a certain (absolute) time. In the above example, HTLCs are used in the following sequence:

- Funds from A are locked in the A-B channel (hash: B or timeout (3T): A)
- Funds from B are locked in the B-C channel (hash: C or timeout (2T): B)
- Funds from C are locked in the C-D channel (hash: D or timeout (1T): C)
- Funds are released to D in the C-D channel after D sends the pre-image to C
- Funds are released to C in the B-C channel after C sends the pre-image to B
- Funds are released to B in the A-B channel after B sends the pre-image to A

Time-outs (1T, 2T, 3T) are selected such that each intermediate node is guaranteed to have some time, after receiving the preimage just in time on the outgoing funds, before the time-out of the incoming funds expires.

1.2 Exchanging crypto-currencies over the Lightning network

It is not necessary for all channels in the Lightning network to consist of the same (crypto-currency) asset. It is even possible for different channels of the same node to be based on different underlying blockchains, provided that the time-out difference between HTLCs of transactions routed between the two is large enough to sufficiently reduce risks created by variability in block creation rates in the two blockchains.

If a node accepts an incoming transaction on one channel that is routed to another channel with a different asset, that node ends up receiving one asset and sending another asset. In other words: that node ends up performing an *exchange*. In the Lightning network, the payer initiates the payment, and determines the route and the amounts on all channels. Therefore, the payer also specifies the effective exchange rate of the transaction. Of course, an intermediate node can always choose to refuse forwarding a transaction if the exchange rate is unacceptable. This is equivalent to refusing a transaction because it pays insufficient fees to the intermediate node, or because the HTLC time-out difference for the intermediate node is insufficient.

An exchange transaction can include multiple exchanging parties: for instance, an ETH-holding customer can buy goods at a BTC-accepting shop with a route an ETH/LTC exchange and a LTC/BTC exchange. For simplicity, we will only consider routes that contain two exchanging parties. On Lightning, this ends up being a payment-to-self route, with one asset being sent from A to B and the other asset being sent from B to A.

2 The market

It is assumed an information exchange mechanism is possible where Lightning nodes can publish exchange offers, to buy or sell assets on one channel in exchange for different assets on another channel. Such offers are informal and not cryptographically binding in any way. This is similar to the gossip protocol of the Lightning network, where nodes publish offers to forward payments at a certain fee rate. In both cases, there can be legitimate reasons to refuse payments, even if they conform to the terms and conditions of an earlier offer: for instance, channels might be saturated because of earlier transactions, or the exchange offer was already fully executed by earlier transactions.

False offers can be identified relatively quickly by trying to route a payment, and observing it being canceled. The offer-accepting party may then decide to blacklist that node for a while, especially if the exchange offer continues to be published while the execution attempt failed.

3 The delay attack

An exchange transaction has two exchanging parties: one intermediate node (B) that placed an offer, and one node (A) that accepts the offer by making a route from A to B using one asset (X), and then back to A using the other asset (Y). Both parties have a step in the Lightning protocol where they can delay the decision on whether to execute or cancel the transaction, up to the time specified in a HTLC time-out. These time-outs can be quite large, because they have to allow operators of intermediate nodes to manually repair/replace anything (including hardware and internet connections) that might break in the middle of ongoing transactions. In the mean time, the exchange rate between X and Y will be changed, either in the advantage of A, or in the advantage of B. In a route A -(X)-> B -(Y)-> A, they can take advantage of creating delays in the following ways:

- After incoming X are locked in, B can delay locking in outgoing Y. Just before A's incoming HTLC time-out is expected, B decides:
  - The value of X has increased w.r.t. Y -> lock in the outgoing Y.
  - The value of X has decreased w.r.t. Y -> cancel the transaction.
- After incoming Y are locked in, A can delay claiming the incoming Y. Just before A's incoming HTLC time-out is expected, A decides:
  - The value of X has increased w.r.t. Y -> cancel the transaction.
  - The value of X has decreased w.r.t. Y -> claim the incoming Y.

Assuming both parties have the same impression of the X/Y exchange rate, if they both attack each other, they will always end up with a canceled transaction. With one attacking and one "honest" (non-attacking) party, the honest party will experience a long delay in transactions, have about half[1] of the transactions canceled, and receive sub-optimal exchange rates for the transactions that do succeed. It is therefore desirable to reduce or eliminate the opportunity for this attack.

4 First countermeasures

The offer-accepting party can protect itself against the offering party in various ways. The offer-accepting party might simply tell the offering party that it will immediately cancel any transaction that doesn't get locked in within a very small time window. This is a realistic threat, since, in the case where the offering party benefits from delaying and then locking the transaction, the offer-accepting party benefits from canceling. The offering party might then conclude that trying the abuse simply isn't worth it.

The offer-accepting party can also protect itself by splitting up the transaction into small chunks. If one chunk is delayed, it can simply stop trusting the offering party, and choose to send the other chunks through other offers, if available at sufficiently attractive prices. Splitting up transactions is recommended anyway, since Lightning works much better for small transactions than for large ones.

Similarly, the offering party can protect itself by only accepting transactions in small chunks; the chunk size to be specified in the offer. Larger incoming transactions can simply be canceled immediately. This does not help in any way unless the offering party can somehow see who is performing transactions: otherwise, a single malicious party can simultaneously create many chunk-sized transactions, rendering the chunk size limit meaningless. Lightning provides

[1]The honest node may change this ratio: for instance, with an exchange rate that is already significantly better for the attacker than the market exchange rate from the start is less likely to be canceled by the attacker.

no native way for the offering party to check the identity of the offer-accepting party, but the offer-accepting party might sign the transaction hash in advance to the offering party, to connect the transaction to a (public key) identity. Again, this doesn't help anything unless there is some kind of PKI and/or reputation system in place. With completely unverified keys, an attacker can simply generate and use a new key pair for every new chunk.

With these measures in place, the offer-accepting party might be sufficiently protected, but protection for the offering party is quite limited. This might discourage market participants from creating offers. In the absence of extra measures, there will probably be a price premium on offers; effectively, a large spread between buy and sell offers.

5 The latency monitoring service

In order to allow a neutral, trusted third party to witness the latency, a couple of new interactions are added between the offer accepting party (A), the offer making party (B) and the trusted latency monitoring party (M) (see figure 1):

- A deposits a certain amount of collateral funds in an account managed by M, for instance with a Lightning transaction.
- Prior to starting the exchange transaction, A sends M a signed message containing the hash, the time-out on B's outgoing funds, and the amount of collateral funds that must be reserved for this transaction.
- On receiving an incoming exchange transaction, B checks whether it corresponds to an open offer. B queries M on how much collateral funds are reserved, and checks whether this amount is sufficient. If this is not the case, B immediately cancels the incoming transaction. If all is OK, B forwards the transaction.
- Normally, the incoming funds of A get locked pretty quickly. If, however, that takes longer than normal, A can inform M that the transaction will be canceled, with a signed Cancel message. A should do this as soon as possible, to prevent loss of collateral funds. Also, after sending this message, A must actually cancel the transaction if it arrives, and never publish the preimage to anyone. On receiving the Cancel message, M will inform B about the cancelation. Note that this scenario is not shown in figure 1.
- If the incoming funds of A are locked in, A can still try to delay the transaction, but that would lead to a loss of collateral funds. It should send the preimage to M as soon as possible, but not before it has successfully claimed its incoming funds.
- On receiving the preimage, M forwards it to B. It is possible for the preimage from M to reach B before its outgoing funds are released. This may

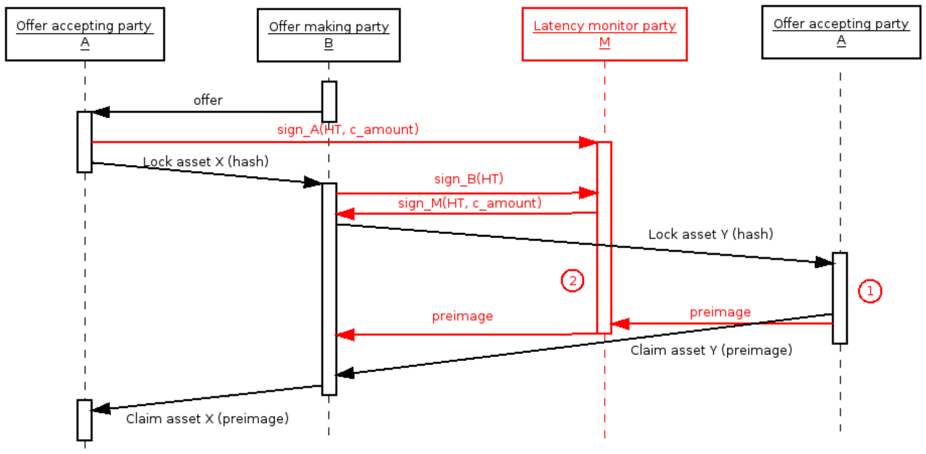

Figure 1: The red lines are added for the monitoring service. In case A delays the normal sequence (1), this adds to the latency registered by M (2).

allow B to claim its incoming funds before it is forced to release its outgoing funds. However, since A has already successfully claimed its incoming funds, the preimage is already known on that part of the route too, and the transaction will also succeed on that part of the route. If this was not the case, A becomes the victim of its own misbehavior.

- If B's outgoing funds are claimed before it has received the preimage from M, it may send the preimage to M. This informs M that the transaction succeeded (after some time).
- If the time-out of B's outgoing funds is exceeded, and M has received neither preimage nor Cancel, it will by default assume the transaction has timed out.
- M records the time it took (starting at the moment A reserved the collateral funds) for A to return the preimage, or to send a Cancel message, or for the transaction to time-out. Based on this time, a certain amount of the collateral funds is assigned to B, and the rest remains accessible to A. The amount assigned to B increases with the latency; on time-out, the full collateral is assigned to B. If both a Cancel message and the preimage are received, the full collateral is assigned to B.
- A and B may withdraw any funds they have at M that are not locked up as collateral for an ongoing transaction, for instance with a Lightning transaction.

This process allows for the following:

- A still has the ability to execute the delay attack, but this will result in an increasing amount of collateral funds being transfered to B.

- In the case of delays in the Lightning network not caused by A, A has the ability to prevent losing collateral funds by immediately sending a Cancel message to M. This also prevents B from deliberately claiming collateral funds by delaying the locking of outgoing asset Y.
- The choice of how much collateral is needed for a certain exchange depends on the perceived volatility and the risk of abuse. It is a subjective choice, and best left to individual market participants.
- A and B do not have to reveal the traded assets and amounts to M. Depending on the policies of M, they may or may not be allowed to trade under pseudonyms.
- B does not have to know any identity information about A. A only has to know the Lightning node ID (a pseudonym) of B. The identity of B may be hidden even more if A and B can negotiate the Lightning route in more advanced ways, where B creates the part of the route surrounding its node.
- M has to hold collateral funds of participants, but these are expected to be much less than the actually traded amounts. If funds are deposited immediately before and withdrawn immediately after the exchange, M only has to be trusted with the funds for the duration of the transaction.
- The rules are defined in such a way that M does not have a profit that depends on the outcome, so that, as long as M does not also act as a market participant, M can be impartial.
- There may be multiple competing latency monitoring parties; B may accept collateral stored at several different parties. There is expected to be some network effect for latency monitoring parties (especially since they need to be trusted), but the barrier to entry in this market is very low.

6 Alternative approaches

6.1 A reputation system

It is possible for the latency monitoring service to, instead of holding collateral funds, publish signed statements about interactions between different identities. These statements can then be used to build up a reputation network between different identities.

Appendix A analyzes the properties of such a reputation system. Compared to collateral funds, a reputation system seems to have serious issues, for which no known solution exists.

6.2 De-centralizing the latency monitoring service

De-centralizing would be difficult, because consensus would be needed on the latency measurement, without having a central authority. A blockchain-based solution has some issues of its own, among which:

- It scales badly. Including a sort of "collateral smart contract" for every transaction would defeat the purpose of the Lightning network.
- The typical time between two blocks (the only de-centralized time-keeping mechanism) is much larger than desirable for keeping the transactions fast.

One way to de-centralize could be to let many different parties each control a small part of the collateral funds. However, that leads to these problems:

- A Sybil attack is possible.
- An attacking latency monitor can simply walk away with the collateral funds.

It all comes down to the fact that trust is hard to de-centralize.

7 Future work

In an ongoing effort to remove Central Points of Control, maybe the downsides of the reputation system approach can be eliminated in the following ways:

- *Proof of Burn* as the cost of reputation, to make sure nobody can get reputation for free. In a sense, this is similar to how the creation of new bitcoins is linked to the destruction of electrical energy.
- Some decentralized way to synchronize the use of reputation in different transactions, to make sure a single amount of reputation is never used simultaneously in multiple on-going transactions.
- A further reduction of the responsibilities of the latency monitoring service; maybe it is sufficient to let it be a (trusted?) cryptographic timestamping service with a low latency and a high time resolution.

References

[1] Joseph Poon and Tadge Dryja. Lightning network, 2015.

A Reputation systems that have pseudonymous participants

A.1 Introduction

In certain systems, it is possible for participants to profit, at the cost of other participants, by breaking the rules they are supposed to follow. One way to counter this is to punish any participant who breaks the rules; the punishment should be sufficiently strong that nobody should consider breaking the rules profitable anymore. However, there are systems where punishment is impractical, for instance because participants are anonymous, and can disappear without a trace, before being punished. An alternative could be that participants keep track of each others' past behavior, to estimate trustworthiness for future encounters. Here, wel will explore the properties of such a reputation system.

A.2 Assumptions

It is assumed participants are known under pseudonyms, for instance as public keys. It is assumed a pseudonym can easily be abandoned, a new pseudonym can easily be created, a participant can easily hold multiple pseudonyms (without other participants being able to discover that those pseudonyms belong to the same participant), and reputation can only be attached to a pseudonym.

It is assumed that a reputation consists of a single scalar value. Without loss of generality, this value can assumed to be zero for newly created pseudonyms, and a higher value can be associated with a better reputation.

A.3 Properties of the system

Under the given assumptions, a reputation system has to have the following properties:

1. All active pseudonyms have a reputation value greater than or equal to zero: if a pseudonym ever receives a negative reputation value, it becomes in the interest of the owner to abandon that pseudonym and continue acting with a new pseudonym, which starts with a zero-valued reputation.

2. Interactions where the worst-case damage would lead to a negative reputation value for any identity must be avoided.

3. Normal, good interactions are beneficial to both participants, and in terms of reputation gained, must be a positive-sum game. If it were a zero-sum game, the long-term reputation gain of honest participants would be zero, and there would be no way to distinguish long-term honest participants from new, possibly non-trustworthy participants. If normal, good interactions were a zero-sum game, long-term honest play would not benefit a participant in any way, and a participant would be better off cheating every time and replacing negative-reputation pseudonyms with new pseudonyms every time.

4. There must be an economic cost associated with building up a reputation value: otherwise, an attacker can benefit by building up a reputation for free, and then performing an attack, based on the built-up reputation. The cost must be at least as high as the expected maximum profit from performing an attack based on the built-up reputation: otherwise, repeatedly building up reputation and performing the attack would be profitable.

5. The reputation value gain from a positive interaction, and the reputation loss from a negative reaction, are proportional to the (estimated) value gained/lost by the counterparty through the interaction. If this were not the case, it would create an incentive to either split up a large interaction into several small ones, or to merge several small transactions into a single large one, in order to improve the total result (reputation + other gains). That way, participants would still be able to accomplish a proportional scale.

6. The reputation gained with a good interaction must be evaluated based on its place in the network of all interactions. An attacker can try to perform a Sybil attack by creating many pseudonyms, and having positive interactions between them; such interactions must not have an impact on how other participants view the reputation of the attacker. The reputation of a group (for instance, the group of all pseudonyms of an attacker), as seen by another group (for instance, the group of all honest participants) must be based on the interactions between the two groups. Identification of different groups must be based on the network of interactions, since there is no other source of information.

7. In case the reputation of identity A is built up by paying the cost of the reputation to identity B, identity C trusts A based on the built-up reputation, but C ends up being attacked by A, then C experiences a loss, while B experiences a gain. An attacker who controls both identity B and C can abuse this to profit from attacks. To prevent this possibility, the identities that profited from building up the reputation of an attacker must lose an amount of reputation that is at least as large (in terms of reputation build-up cost) as the profit the attacker received from the attack.

8. There must be a mechanism to prevent that a single reputation value can be used to perform many attacks simultaneously: otherwise, an attacker can invest once the cost of building up reputation, and then take many times the profits from attacks, making the attacks profitable.

A.4 Reputation versus a collateral

The properties listed in the previous section paint a picture of reputation that is similar to collateral funds: you can "deposit" into your "reputation account" by performing good interactions (which involve a financial cost from your side), and you can "withdraw" from it by performing an attack (which may benefit you financially). Possible differences include:

- A reputation system may deliberately include an inefficiency (performing an attack will result in gains that are (much) lower than the costs of the corresponding build-up of reputation). However, the same may be done for a collateral funds system.

- In a collateral funds system, the collateral funds can be used to compensate the victim of an attack. In a reputation system, the “deposit” may happen in an interaction with a different participant than the “withdraw”, in which case the victim doesn’t get compensated for the loss.

- A collateral fund, managed by a trusted third party, has a simple way to prevent it being used simultaneously for multiple transactions: the trusted third party can simply keep track of this. For a decentralized reputation system, there is no obvious way to prevent a built-up reputation from being used for multiple simultaneous attacks.