Beam Lightning Network Position Paper

Fabian Jahr, for Beam

February 12, 2019
V 1.0

1 What is the Lightning Network?

The Lightning Network is a second layer scaling solution for blockchains. It enables extremely fast and cheap transfers of value off-chain that can be settled on-chain at any moment. At the same time, the Lightning Network does not sacrifice trustlessness, security or decentralization of the underlying protocol. To achieve this, the Lightning Network is based on the broader concept of payment channels, which are implemented through a limited set of smart contracts.

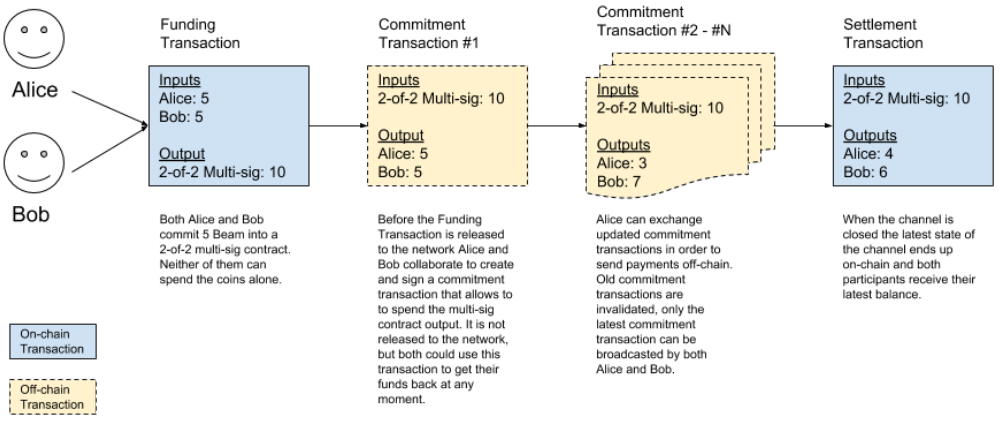

Figure 1: High level mechanics of a payment channel between Alice and Bob. Inspired by figure 12-4 of Mastering Bitcoin.

Off-chain payment channels are not a new idea. The concept was already a topic of discussion in the Bitcoin community when Satoshi was still around, although his original idea of payment channels was flawed [1]. Several ways of constructing payment channels were explored by researchers over the year until Joseph Poon and Thaddeus Dryja presented their version of payment channels, called the Lightning Network, in 2015 [2]. Their findings, along with the support of several other researchers, was the breakthrough that led to the state of the Lightning Network that we can use on Bitcoin mainnet today.

At the time of writing the Lightning Network on Bitcoin mainnet is counting 5,700 active nodes, 22,000 public channels and a capacity of over 600 Bitcoin (2M USD) [3]. As the network only got started little more than a year ago, these numbers are clearly impressive. While there are still challenges ahead for the Lightning Network itself, especially making it more robust and user-friendly, the Team of Beam has built a strong belief that

Lighting Network has already proven to be the currently best choice of a scaling solution for blockchains today and thus worth a significant investment of our resources.

2 Why Lightning Network on Beam?

At first glance users may ask if a scaling solution is really necessary on Beam: after all, Beam has a block time of only one minute while Bitcoin only forms a block only every 10 minutes. Why is there still the need for a Lightning Network if Beam is already "ten times faster"?

To begin with, faster block time does not necessarily mean faster payment confirmation. While in Bitcoin it is generally advised to wait for 6 confirmations (6 blocks mined on top of the transaction) for a serious transaction to be considered fully settled, users of Beam may rather want to wait for a period in the realm of up to 60 confirmations to get the same level of confidence in the received payment, especially while the network is still young and hashpower is comparably low.

Furthermore, there are many use cases that we want to use digital cash for, where there is only minimal tolerance for wait times. It may be ok to wait for a couple of minutes when you are paying for beers at a local bar and you are not in a hurry to leave. But think of just quickly grabbing a drink from a vending machine while you are trying to catch your train or paying at a busy checkout with a long queue of other nervous customers behind you. For Beam to take over these use cases as well, we clearly need to achieve "Visa level" speed of payments, if not even beat it. Trials have shown that the Lightning Network is able to achieve payment speeds that make the payment experience as quick and easy as a credit card payment which make it one of the cornerstones along the way to achieve mainstream adoption of cryptocurrencies in general.

In addition, Beam's transaction capacity per second is currently about three times that of Bitcoin, which is a nice increase. But it is still not nearly at a level that would allow for "Visa level" transaction throughput. If we can achieve a user experience that enables mainstream adoption as described in the previous paragraph, we also need the transaction capacity that is able to carry it. A second layer solution like Lightning Network allows transaction capacity to scale orders of magnitude without the need to extend the base layer of the blockchain. Mainstream adoption would simply not be feasible without it.

Last but not least, we are mitigating issues that are not a problem for Beam at the moment but could become more important in the future just as they have been for Bitcoin in the past. Fees on the Lightning Network should be significantly lower than using on-chain transactions. While fees are low in Beam at the moment this may change as we see more significant adoption. The same goes for block space: Beam blocks are far from being full right now but will fill up further as we see more transactions.

3 Lightning Network challenges on Beam

There are many differences between Mimblewimble and Bitcoin, and some of these differences are posing issues for the Lightning Network integration of Beam. Some, like the interactive nature of transaction construction, luckily solve themselves. Lightning nodes need to be reliably online most of the time anyway. The major issue, however, is the lack of scripts in the Mimblewimble protocol.

This is where the magic of scriptless scripts comes in, a concept introduced by Andrew Poelstra along with his initial thoughts on Mimblewimble itself [4]. Scriptless scripts allow for the construction of smart contracts without the usage of a scripting language by encoding the script into a signature instead. They are blockchain-agnostic, the only major requirement is the use of Schnorr signatures. Scriptless scripts are a part of Beam today and

as such have already provided solutions for many of the requirements that the Lightning Network is built on.

Lightning Network funding transactions are essentially a 2-of-2 multi-sig transaction, which is possible in Beam through the collaborative construction of the transaction demonstrated in our documentation as part of Atomic Swaps for example [5]. The main difference is that the number of messages going back and forth between the peers is slightly increased due to the difference in the underlying Mimblewimble protocol.

The construction of Hash Time Locked Contracts, the core concept of the Lightning Network, requires two more smart contract schemes: hashlocks and relative timelocks. Hashlocks have been a part of the base protocol of Beam from the start [6]. But while Beam has implemented absolute timelocks, relative timelocks are not available yet. Thankfully, members of the Mimblewimble Mailing List, mainly consisting of members of the Grin development team, have identified a solution to this issue already which we agree with and plan to implement in Beam as well [7].

But how do we chain these concepts together without a script? Where there are multiple execution paths of a script in Bitcoin, in Mimblewimble we will need to use multiple transactions that build on top of each other. The spending requirements of these transactions will work together to form the same construct as HTLCs in Bitcoin and the participants of a payment channel will collaborate to construct these different transactions. This, again, leads to an increase in messages being sent between the participants of a channel, but the gain is more than worth it.

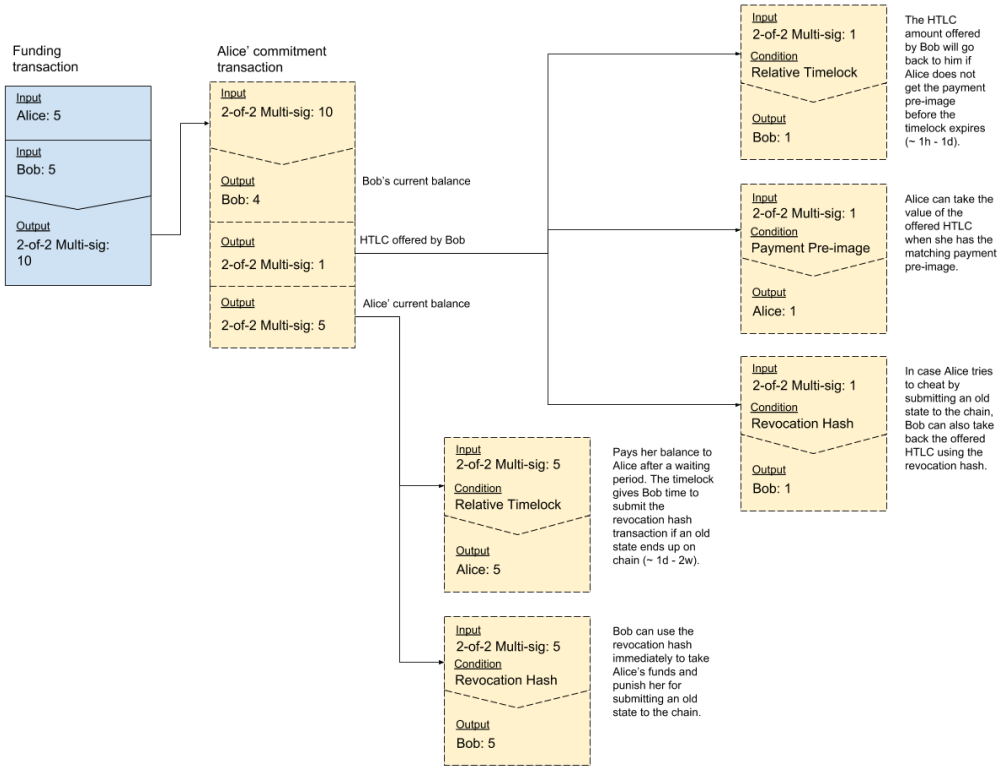

Figure 2: Exemplary transaction structure on Alice' side of a Lightning Network payment channel on Mimblewimble.

4 Mimblewimble's advantage over Bitcoin

Its privacy properties are one of the major reasons why Mimblewimble chains are so appealing to many people. This feature also shines when comparing Lightning Network implementations on Bitcoin with its still theoretical counterpart on Mimblewimble chains. A funding transaction that is opening a channel on the Bitcoin blockchain followed by a closing transaction that is resolving the same channel together leave a pattern of transactions that can be fairly easy to spot. At the very least, the two participants of the channel, the initial state, and the end state are exposed and could be combined with other data from monitoring the Lightning Network itself to further invade user privacy. Mimblewimble, on the other hand, uses confidential transactions which hide participants and amounts on-chain, making these kinds of attacks impossible.

5 Project Roadmap

Throughout the past couple of weeks, the development team of Beam has been researching the necessary requirements for a Lightning Network integration with Beam. For that purpose, we are working on a fork of the Lightning Network RFCs that document the adaptations to the protocol on a high level to work with any Mimblewimble implementation [8].

For Beam specifically, the next step is to implement relative timelocks. On the core protocol, this is the last step necessary to lay the groundwork for a Lightning Network integration. Furthermore, the team plans to evaluate current Lightning Network implementations for the possibility of a fork. Several implementations are at an impressive condition and the team would be happy to gain a speed boost from taking over the work that has already been done. That goes especially for the parts that require minimal adaption for Mimblewimble, like the transportation layer and onion routing, to name a few examples. Conversely, the Beam development team, of course, hopes to then be able to contribute back to the source project once it is up to speed with the code base and thus contribute to the Lightning Network community not just on Mimblewimble but overall.

References

[1] Bitcoin.it Wiki. *Payment channels*. URL: https://en.bitcoin.it/wiki/Payment_channels.

[2] Joseph Poon and Thaddeus Dryja. *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. URL: https://lightning.network/lightning-network-paper.pdf.

[3] 1ML. *Real-Time Lightning Network Statistics*. URL: https://1ml.com/statistics.

[4] Andrew Poelstra. *Mimblewimble and Scriptless Scripts*. URL: https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-06-iheie-paris/slides.pdf.

[5] Beam Github Wiki. *Atomic swap*. URL: https://github.com/BeamMW/Beam/wiki/Atomic-swap.

[6] Beam Github Wiki. *Core Transaction Elements*. URL: https://github.com/BeamMW/Beam/wiki/Core-transaction-elements.

[7] Grin Github Wiki. *Contracts*. URL: https://github.com/mimblewimble/grin/blob/master/doc/contracts.md#relative-timelocked-transactions.

[8] Fabian Jahr. *Lightning Network on MimbleWimble Specs*. URL: https://github.com/fjahr/lightning-mw.