Lumino Transaction Compression Protocol (LTCP)

Sergio Demian Lerner, Chief Scientist RSK Labs

Date: February 24th, 2017

REV: 10

Abstract

In this paper we present the Lumino Transaction Compression Protocol (LTCP), a key part of the Lumino Network, the RSK's off-chain payments network. The LTCP protocol allows a higher volume of on-chain of transactions than traditional blockchains for the same blockchain space consumed. The LTCP protocol is an improvement to the RSK platform for reaching 2000 on-chain tps. However, LTCP can also be implemented as a soft-fork to Bitcoin, increasing Bitcoin throughput to 100 tps or even more, depending on the usage pattern. The LTCP protocol could enable the Lumino Network to reach one billion active users.

Index

Foreword

As the Internet did with information, Bitcoin and cryptocurrencies can become the cornerstone to build a new kind of Internet. An Internet that would not only be suitable for knowledge sharing but also for value transfer. That value might be in the shape of property titles, stocks, money or other tokens such as voting rights for an autonomous organization. The creation of this Internet of Value combined with the growing smartphone adoption could turn financial inclusion into a reality. But for this technology to reach billions of people it needs to be cheap and it needs to be decentralized. RSK platform is a decentralized Bitcoin sidechain. RSK technical vision of the Internet of Value comprises multiple protocol layers, each layer solving a different need for the transfer of value, together forming a coherent yet flexible protocol stack. This paper introduces the Lumino Transaction Compression Protocol, a key part of RSK protocol stack.

Introduction

The field of cryptocurrency, or decentralized digital cash, is in its infancy. Before 2009, most attempts to create digital currencies required trusted third parties. All digital payment networks were also centralized. In 2008, Bitcoin, the first decentralized cryptocurrency and payment network, was invented, and launched the following year. However, Bitcoin's decentralization and censorship resistance were tradeoffs for scalability. Different techniques, such as compact blocks, better routing, header first propagation, probabilistic verification, fraud proofs, blockchain pruning, partitioning and payment channels were invented. The culmination of such chain of improvements is the Lightning network. The Lightning network promises to reduce the resources required by each node of the network and enable higher transaction throughputs. However, to reach billion of users, the Lightning network requires also a high volume of on-chain transactions. This happens every time a hub misbehaves or when one party disappears. Also users may need to top up their payment channels often, requiring more on-chain transactions. Assuming each user tops up a main payment channel every month, and each channel is also settled every month, and there are no significant misbehaving nodes, Bitcoin throughput would enable 2M users to use payment channels. The LCTP protocol presented in this paper allows top ups to consume as low as 5 bytes in the blockchain, enabling one billion users to use the RSK platform.

Our Contribution

The RSK platform follows similar ledger structure than Ethereum: it's based on accounts rather than UTXOs. Each user controls one or more accounts. Each account is controlled by one private

key of the a ECDSA scheme. The public key is mapped to a shorter RSK address. Transactions can deduct funds from an account, and transfer them to other accounts or smart-contracts. Each transaction is signed and the account from where the funds are taken is extracted from the signature by the ECDSA public key recovery procedure.

In this paper we present a technique for transaction compression that allows processing a higher volume of transactions but storing much less information. We also present the basics of the Lumino network, a lightning-like extension of the RSK platform that uses the LTCP protocol.

Lumino Transaction Compression Protocol (LTCP)

The critical insight behind Lumino compression is delta compression of selected fields, from a previous referenced transaction and the aggregate signing of previous transactions, so previous signatures can be disposed. Delta compression is done by allowing each transaction to refer to a previous transaction from the same owner, which is used as a template. Any field can be overridden, and unmodified fields are copied intact. A transaction that uses LTCP contains several fields, some of them optional and some of them are persistent. Persistent means that they will become part of the blockchain forever, while non-persistent fields may or may not become part, depending on future transactions. These are the user provided transaction fields:

- **nonce:** a nonce [persistent if repeated]
- **seqNum**: a sequence number [optional,persistent]
- **amount**: amount of funds to transfer [optional,persistent]
- **receiver**: the address of the receiver [optional,persistent]
- **deltaTxNonce**: this value specifies the nonce of the reference transaction for delta compression[optional,persistent]. If missing, is specifies the previous nonce.
- **gasPrice** (or fee): an amount of fee to pay [optional, persistent]
- **gasLimit**: max number of steps to execute [optional, persistent]
- **data**: arbitrary user data to be sent to the receiver (used mainly for smart contracts) [optional, persistent]

The set of persistent fields in a transaction is called the Persistent Transaction Information (PTI). An additional auto-computed field is added to the PTI, the **accountIndex.** This is an increasing sequence number assigned to each new account created. What is actually signed in a transaction is not the user provided transaction data, but a compound record with additional information, called the **SigRec**. The SigRec contains all fields, plus the additional field:

- **prevTx**: a hash of the SigRec of the previous transaction from the same source account.

The SigRec contains all the fields in a fixed order. To sign a transaction, its corresponding SigRec is hashed, and the hash digest is signed with the ECDSA private key corresponding to the

sender's account. Amount is a big integer represented by a base-10 exponent (5 bits) and a mantissa (variable number of bits), allowing greater compression. A 4-byte accountIndex can support more than one billion accounts. Also it's possible to replace the accountIndex with a shorter field, representing the block number delta and transaction index in the block of the previous transaction from the same account in the blockchain, using compact variable-length integers. For instance, if the prior transaction is the second of the previous block, the value (-1,2) would encode the reference, consuming no more than 2 bytes.

A transaction is accepted if it satisfies standard conditions (enough funds, well formatted, etc.) plus additional conditions on the sequence number, which allows transaction replacement for the Lumino Network. The transaction fields that are missing from the user provided fields are copied with the fields of the delta transaction. The exact conditions to accept a transaction are the following:

- the nonce must be higher or equal than the nonce of the previous transaction included in a block from this account
- If the nonce is equal, the seqNum value must be higher than the predecessor.
- If the nonce is equal, it is included in the PTI to signal it.
- If present, deltaTxNonce must be specify a transaction at a distance lower than a value D (replacements included in the block counted in the distance).
- prevTx = hash (prevSigRec) where prevSigRec is the SigRec of the previous transaction included.
- the source account must have enough funds for the payment.
- the block distance of the delta compression chain must be lower or equal than M.

Block Format

A Lumino block stores two Merkle trees (or Merkle Tries). The first containing all PTIs. The second contains all transaction ids. Transaction ids are built as a hash of the signed SigRec. This second tree is conceptually similar to the segwit witness tree, so we will call all fields that are implicit plus the signature, the witness part.

Disposal

When the witness part of a transaction T is stored in a block we say T is **persisted**. When only the PTI of a transaction T is stored in a block but the PTI of T is referenced via prevTx references by a following persisted transaction T2 in the best chain, and T2 is at a block distance lower than M from T, we say the transaction T is **docked**. The witness part of a docked transaction can be disposed and only the PTI needs to be maintained. Note that a persisted transaction can cease to be so and become docked.

Validity of Blocks

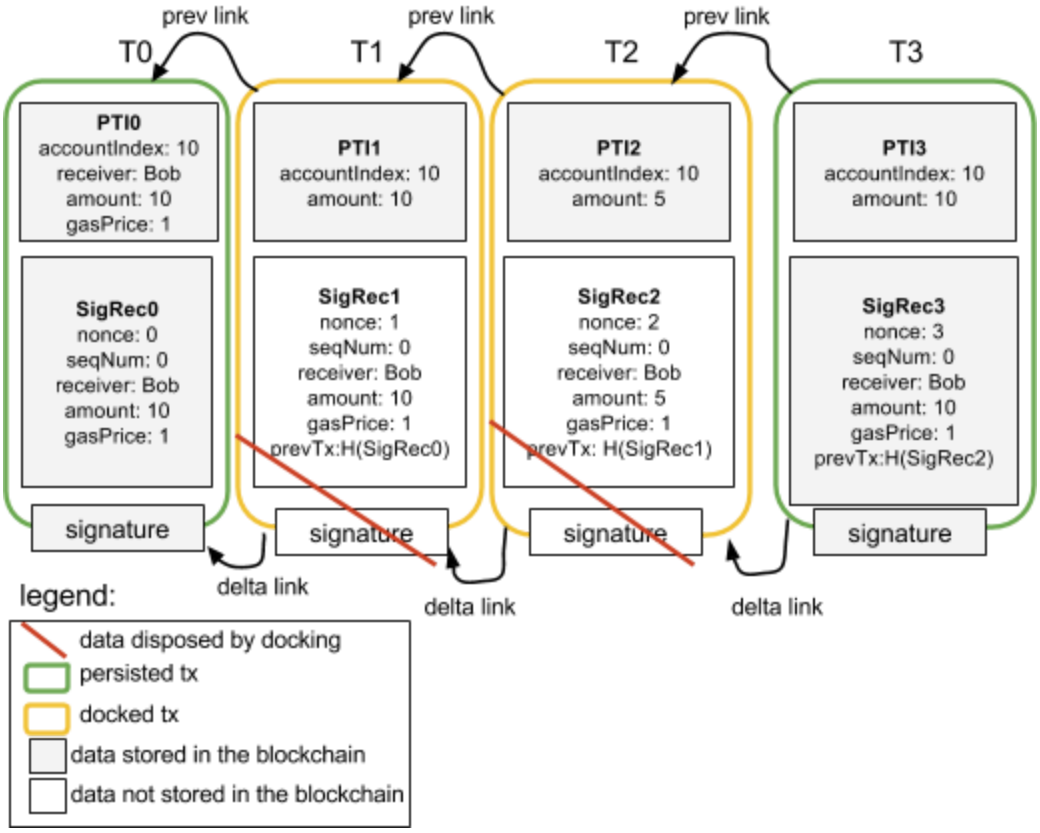
Apart from standard block validation rules (difficulty, parent, etc.), a block is considered invalid if it contains at least one transaction that is neither docked nor persisted.

A block is invalid if it contains a persisted transaction T whose signature fails verification. However, a block is not invalid if it contains a docked transaction that fails signature verification, because no node is obliged to store the witness information. This means that miners can build a

block with “invalid” transactions if they provide correct aggregate transaction signatures that dock the previously invalid transactions in following blocks.

Of course, no full node will accept a block containing invalid transactions, and therefore these malicious miners will need to build a private chain and the publish this chain. Full nodes must however be careful to allow blocks that were previously rejected because of invalid signatures to be re-accepted in the future if they are confirmed by other valid blocks that dock them. It’s preferable to cache a temporal record for a rejected block which lists the invalid transactions that need to be docked.

The first transaction (nonce=0) must have its signature stored forever, to set the sender account address, which is implicit in the signature by signature recovery. The following figure shows an example transaction chain with delta compression, such as in an on-chain payment channel:



Payment Channels

Two parties can engage in an on-chain payment channel by making use of transaction compression. The sender creates one payment to a receiver, and then afterward omits the receiver field from the transaction. If the gasPrice does not change, each following compressed transaction only persist the amount. Generally, the amount requires not more than 4 bytes of storage.

Blockchain Sync

Syncing the blockchain with LTCP transactions is a two-stage process. Whenever a node receives a block, it must try to verify it and find out if there are invalid or missing witness parts. If the block received is old, the node can request following blocks, until all invalid or missing witnesses are overridden by valid counterparts and therefore they become irrelevant. If still some required witnesses are unavailable, then the branch in question cannot become the best branch. If the block received is new, then the node will request all witnesses from it.

Other Enhancements

The Lumino network is a Lightning-style network that uses the LTCP protocol. As the Lightning requires pre-locking of funds and it's controlled by two signatures, we need a new type of account controlled by two signatures. One possibility is that two transaction signatures are passed through the public key recovery protocol, and their public keys are concatenated in a specific order before hashed to obtain the account address.

An alternative is to create a new type of code to be attached to contracts, the "signature validation code", which allows a transaction to originate from a smart-contract, and consume fees from that contract, without involving a second "origin" ECDSA account. Transactions originating from a smart contract would require a sender field. The first transaction (nonce=0) of a smart-contract does not need to store the signatures forever, since their public keys are explicitly specified in the signature validation code.

In both cases the account will be a smart-contract holding the locked funds. This allows settlement accounts to originate from the smart-contract, and therefore their signatures can be disposed after the next settlement has been included, reducing the storage requirement for settlements to as low as 6 bytes, depending on how the smart-contract encodes and compresses balances. The SeqNum field is used with the same nonce value allows to update channel balances, but prevent old transaction to supersede newer ones. This is necessary to avoid a former transactions to prevent a later to be included in a block, when both have the same nonce. An alternative is to allow gaps in transaction nonce series, but may lead to miners skipping transactions, and complicates the predictability of smart-contract calls.

Lumino as a Bitcoin soft-fork

The Lumino network is designed for blockchains with fast block rates and account based ledgers. To implement the Lumino Network as a Bitcoin soft-fork, we need to simulate an account based ledger: we need to create an account address space for a sidechain embedded in the Bitcoin block, as an extension block. For each account, an exodus address is created in the mainchain. This exodus address will be a anyone-can-spend address prior sof-fork and a special address post soft-fork. For instance, it can be a push-only scriptPub "<Lumino-tag> <Lumino-address>". To load a Lumino account with bitcoins, those bitcoins will need to be first sent to a Lumino

address, where it will be stored until sent back by a special transaction that consumes from the same anyone-can-spend addresses, providing a signature for the Lumino-address.

Once the funds are on the Lumino network, they can use Lumino transactions or participate in Lumino payment channels. However, the Lumino payment confirmation is still restricted by Bitcoin 10 minute average block interval. To extend Bitcoins header with the fields required by Lumino, the fields can be stored in an OP_RETURN output of the coinbase transaction.

The implementation as a soft-fork also requires an authenticated tree embedded in each block for describing account movements (or transaction receipts), to allow SPV nodes to detect them without having to deal with LTCP transactions.

Scalability of LTCP as Bitcoin soft-fork

If M is equivalent to a year of blocks, and we assume a user performs 2 payment a day, then the user performed 730 payments. Let D be equivalent to a 10 transactions. All those payments can be compressed. A realistic assumption is that 10% will be performed to random addresses, while the remaining 90% will be recurring payments to 73 different addresses.

The 10% of random payments will consume on-chain 20 (receiver) + 4 (amount) + prev tx link (3 bytes) = 27 bytes, totalling 2 Kbytes.

The 90% recurring payments will consume on-chain amounts (totalling 4 Kbytes), plus 73 addresses (1.5 Kbytes).

We assume all delta references are available within the D depth restriction. Therefore the 730 transactions consume 7.5 Kbytes, or 10 bytes per transaction. However full nodes that are online still have to process all transactions, and store them.

Finally, Lumino as a soft-fork does not benefits from Segwit space savings, since Lumino exodus address is an anyone-can-spend output, the transaction that consumes this output specifies an empty script.

Scalability

To see how many users and transactions/second Lumino can serve, we must analyze possible values for M and D.

The value M should be limit to reduce the need of block look ahead. For a proof-of-work based chain, this is generally not a problem, since each block carries PoW, a denial of service attacks by adding a high number of confirmations to an invalid block (with a transaction without future signature). We can set M to be 1 year of blocks.

The value D must be chosen so that nodes can store the transactions of the last D PTI for each source account, and that those transactions are stored in fast persistent memory (such as RAM or SSD). Suppose D=10. If the average PTI is 20 bytes, each source account consumes 200 bytes. Assuming the LTCP protocol uses 100 GB of SSD space for temporary storage, the network can

handle 500M users (this figure does not take into account the Lumino Network, just the blockchain layer).

However, to reduce the time required to sync after a blockchain reorganization, nodes may store the full transactions in the last B blocks (e.g. B=20). If the blockchain block interval is 10 seconds, and a block contains a maximum of 30K transactions (3K tps), then a node needs to store in cache 60 Mb of transactions, which is negligible. Miners may store full transactions more more time, such as a month, to allow the network to reorganize the network after a long split.

Assuming an Internet connection that allows 0.5 Mbyte/sec transfers (4 Mbit/sec, 42.5 GBytes/day)[1], and assuming blocks propagate using "compact blocks" or similar block compression technologies, a node supports 7K tps using the full bandwidth. Using one third of the bandwidth seems reasonable for a home PC, achieving 2.3K tps. If each user performs 2 payments a day, then the network can serve 100M users.

A modern home PC CPU can verify 8000 ECDSA signatures per second, so CPU is not the bottleneck.

As a comparison, if Bitcoin was allowed to use the same bandwidth, Bitcoin blocks would be 100 Mbytes, and Bitcoin throughput would be about 550 transactions/second. However the Bitcoin blockchain will occupy 40 times more than a Lumino blockchain, yet enabling approximately one fourth of the transaction volume.

Predicting the number of active users that the Lumino network can handle it's not easy. Transaction compression can enable fixed amount top ups using only 6 bytes on the blockchain. However opening the payment channel would require constant amount of bytes and the last transaction containing signatures also will consume a fixed size per channel. Assuming each payment channel consumes a minimum of 150 bytes, and each user tops up and settles the payment channel once a month, the Lumino network can serve 1 billion active users in 4 years, as a decentralized network based on the the hardware resources expected to be available for home PCs at that time (mainly higher capacity SSD drives).

Summary

Lumino transaction compression is an alternative for scaling a blockchain that achieves high compression ratios. Transaction compression means data reuse, which means that Lumino higher compression is a tradeoff for lower privacy. However, the tradeoff choice is left to individual users. The LTCP protocol enables storage-compression but has resource bottlenecks: bandwidth usage and, to a lower extent, CPU usage. However, if a node only receives and verifies a fraction of the LTCP transactions, they can be informed of transactions with invalid witness data easily since LTCP transactions allow short fraud proofs, so sharding is possible.

[1] This has been confirmed by the research of Adem Efe Gencer and Emin Gün Sirer in http://hackingdistributed.com/2017/02/15/state-of-the-bitcoin-network/ (5 Mbit/sec average)

LTCP forms the transaction layer to the Lumino network, RSK's off-chain payment channel network, which aims to enable the reach of billions of active RSK users.