Layer One dot Five

Lightning-compatible Stablecoin Backed by Bitcoin Collateral

Miro Sarbaev Lexis Tikhvinskiy

April 29, 2022
Commit 7b8f67c

Abstract

Layer One-dot-Five (L15) is a system that enables L15$ - a stablecoin, soft-pegged to the US dollar and backed by Bitcoin collateral. L15 combines the principles of the Lightning network and the sidechain: it encumbers Bitcoin collateral in a Lightning-style DLC while its commitment signatures are recorded on the sidechain. L15 is merge-mined with Bitcoin, which allows miners who win blocks on both chains simultaneously to trustlessly validate collateral release. The resulting system is interoperable with the Lightning Network.

Along with enabling the stablecoin, L15 has the potential to be a platform that developers can use to implement a subset of smart contracts, covering practical consumer finance use cases, like pull transactions and subscription payments.

For the latest version of this paper please visit
https://arrowlabs.com/L15-stable.pdf

Table of Contents

1 Introduction

Bitcoin is gaining recognition as a value-preserving "digital gold." This elevates the importance of having an integrated transactional coin within the Bitcoin ecosystem. A fiat-pegged digital asset that one can borrow using Bitcoin as collateral is a sound approach to increasing the ecosystem's money velocity. While an Ethereum-centric solution is available in the ERC20 ecosystem and could be used by Bitcoin holders via WBTC, most users find this path too expensive. This cost-assessment is made not only in terms of the fees but also considering the time needed to traverse layers of intermediaries, the compounded risk of several innovative solutions chained together, and the added exposure to centralization risks introduced by some intermediaries.

L15 (Layer One-dot-Five) is a system that enables L15$ - a collateralized stablecoin, near-native to the Bitcoin ecosystem. L15 is not strictly *native* since it has its own chain – a Bitcoin merge-mined sidechain that tracks two new digital assets: a stablecoin and a stabilization coin. It is interoperable with the Lightning Network – one can send Bitcoin collateral into L15 directly from a Lightning channel and receive stablecoin into a familiar BOLT-style state channel, provided the Lightning implementation supports PTLC.

When coming up with L15 design, one of the key goals was to uphold a degree of trustlessness and decentralization, worthy of the Bitcoin standard. As a result, L15 uses distributed consensus across the board. We use the term *distributed consensus*, meaning *"a consensus (i.e. global agreement) between many mutually-distrusting parties who lack identities and were not necessarily present at the time of system set up."* (Poelstra, 2015, March 22)

At its core, L15 is a decentralized protocol that utilizes Layer 2 state channel principles while storing channels' states in the L15 blockchained ledger, which creates a potential for implementation of financial smart contracts.

While leaving the platform possibilities intact, we focus on building a single canned feature on top of L15 - the stablecoin L15$, created in L15 as a product of overcollateralized loan facilitated by the L15 contract. Borrowers deposit collateral in Bitcoin and borrow against their collateral value. If the value of the Bitcoin deposit falls below the minimum collateralization ratio, the loan can be liquidated by the contract. On the other end, repaying the loan is equivalent to providing proof of burn for borrowed L15$ and accrued interest, which automatically makes the collateral spendable by the borrower. As a result, the US dollar value of L15$ is backed by the US dollar value of the underlying Bitcoin collateral held by L15 contracts.

To support the stablecoin function, L15 should be able to maintain the L15$

soft-peg to USD. For that, L15 uses its native Stabilization Reserve coin - L15SR. This coin is used to reward miners and to pay most transaction fees. After all the costs, the excess loan interest is exchanged into L15SR, and these sale proceeds are then burned. When it is necessary to increase the price of L15$, L15SR is minted and sold for L15$, and such sale proceeds are burned. Fine-tuning of the L15$ price is done by adjusting loan interest rates.[1]

Maintaining stability of the L15$-USD pair is a function entirely supported within the L15 chain – it does not require the use of Bitcoin Script. Even though it is a complex financial process, the engineering task of implementing it on a new chain is not as challenging as building a trustless interaction between Bitcoin and L15. In the present paper, we focus on the latter, covering both trust model and functionality of locking and unlocking of the Bitcoin collateral, and how these actions are connected to events in the L15 chain. The emphasis will be on the collateral unlocking since L15 provides a stablecoin-specific solution for a challenge outlined by the inventors of pegged sidechains:

> One of the challenges in deploying pegged sidechains is that Bitcoin script is currently not expressive enough to encode the verification rules for an SPV proof. (Back et al., 2014, October 22)

[1] L15$ economy model is inspired by MakerDAO's DAI (*Dai (Cryptocurrency)*, 2021, August 26).

2 Terminology

Table 1: Terminology

Term	Explanation
L15channel	An L15lightning channel (formed according to the L15 specification).
L15full	L15node that acts on behalf of itself and does not participate in the creation of L15vhive. L15full is similar to a Bitcoin's full node.
L15ledger	An L15 public ledger.
L15lightning	A modified Lightning protocol capable of supporting L15.
L15miner	L15node that participates in the creation of L15vhive by mining new L15ledger blocks.
L15node	Any node on the L15 network.
L15signer	L15node that participates in creating L15vhive by engaging in threshold multisig to hold L15vhive's ends of L15lightning channels.
L15vhive	A collective of nodes working together to produce L15 functionality. L15vhive maintains itself as a single entity in the L15 network by means of L15miner nodes and L15signer nodes having a shared public ledger - L15ledger.
L15vroutnode	A group of L15signer nodes that collectively sign a Lightning channel on the L15 side. L15vroutnode is a part of L15vhive.
At-risk Liquidation	Termination of loan contract due to loan collaterization dropping below the minimum for the contract.
Negotiated Termination	Termination of a collateralized loan contract due to fulfillment of pre-negotiated conditions.

3 Trust Model

L15 uses a UTXO ledger (L15ledger) to track L15$ and L15SR. It is secured via proof-of-work by merged mining with Bitcoin and the recoding of anchors into the Bitcoin chain. L15 blockchain aims at a 60-second block time.

Merged mining is also used for cross-chain validation of pending collateral release CETs[2]. Due to the fact that these CETs purposefully contain outputs that are sized below the Bitcoin dust limit, they can be included in the Bitcoin blockchain only by L15miners who happen to have won the right to record a Bitcoin block.

Additionally, L15 relies on pricing oracles, Olivia(s), who publish the BTCUSD ratio. The oracles (Guillyr et al., 2020, November 03) and multi-oracle support (Cohen, 2021, May 07) are assumed to conform to Suredbits specifications.

Proof-of-work consensus is stateless by nature. Its pristine form cannot store persistent secrets required by L15 for collateral transactions that utilize DLC[3] and state channels. To keep the contract state record and carry out appropriate actions based on the miner's validation decisions, L15 introduces two new groups of anonymous actors - L15signers and L15arbiters. When a user opens a new state channel (L15channel), a dedicated group of L15signers (L15vroutnode) is assembled out of the pool of all signers. L15signers are required to supply bonds in Bitcoin. The selection of an L15signer for an L15vroutenode is probabilistically random, with the random component provided by entropy from the L15miners, while probability depends on the total size of the bond. L15vroutnode acts as a single signing entity, where participating signers use Schnorr threshold multisig to build channel and DLC signatures. L15arbiters are similar to L15signers in how they are selected for signature participation, but their bonds are in L15SR. Acting as a signing entity for miners, they execute rollback actions for contracts, if necessary. Also, if needed, they act on fail-safe scenarios.

An arrangement with L15signers and L15arbiters is reminiscent of a Proof of Stake algorithm. However, an essential difference precludes L15 from being impacted by the PoS deficiencies described in (Poelstra, 2015, March 22): signers and arbiters do not participate in creating the chain finality.

[2]CET is Contract Execution Transaction as described in the DLC foundational paper (Dryja, 2017).

[3]While keeping close to the description of DLC by the inventors (Dryja, 2017), we are using the concept and the term in a non-standard way on a few occasions.

4 Contract

4.1 Parameters

A key contract parameter is loan collateralization, which depends on the current Bitcoin price and total interest accrued by the loan:

$$PCL = \frac{CX \cdot PBU}{CY + TPR} \tag{1}$$

where the variables are:

Table 2: Collateralization Formula Variables

Parameter	Unit	Definition
PCL	Percent	Current Collaterization
TPR	L15$	Total interest acquired
PBU	USD	Current price of Bitcoin in USD

The following Table 3 lists contract constants.

Table 3: Contract Parameters

Param.	Unit	Definition	Note
CX	BTC	Collateral Amount	
CY	L15$	Loan amount	
$cAPR$	Percent of CY, in L15$	Contract Annual Interest Rate	
CTF	Percent of CX, in L15SR	Collateral Termination Fee	Applicable in case of At-risk Liquidation and in case of Negotiated Termination.
CLP	Percent of CX, in L15SR	Collateral Liquidation Penalty	Applicable in case of At-risk Liquidation.
CLD	Time Units	Contract Duration	Time terms are set in block height and expected to be converted to days and months on the client side. L15 does not support contracts unlimited duration, but a contract can be reinstantiated at any time.
CLM	Percent	Minimum Collaterization	Liquidation is triggered by PCL dropping below this value
CLE	Percent	Efficient Collaterization	If PCL is below this value, Negotiated Termination is not possible.
CRF	Fixed in L15SR	Contract reinstantiation fee	At contract reinstantiation, all parameters listed in this table can change.
CIG	Time Units	Contract Interest Granularity	Total interest is recalculated after each time period that equals to CIG.

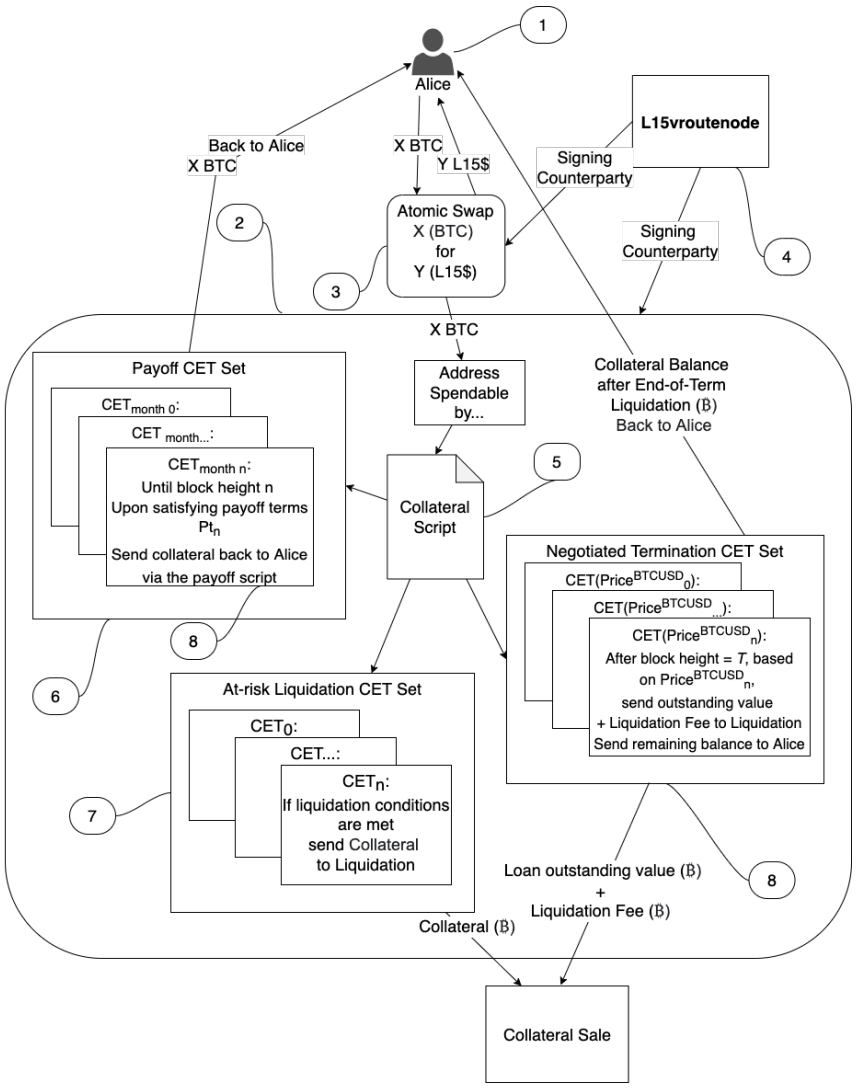

Figure 1: Collateral

4.2 Collateral DLC

A borrower, Alice (fig. 1, 1), enters into a loan contract. She sends her collateral of X Bitcoin to a DLC (fig. 1, 2) via an atomic swap (fig. 1, 3) that pays Alice back Y L15$. Both atomic swap and DLC are countersigned by an L15vroutenode (fig. 1, 4).

For the period up to the contract duration, Alice's collateral is stored in the DLC and is spendable via Collateral Script (fig. 1, 5):

- by the Payoff set of CETs (fig. 1, 6) or
- by the At-risk Liquidation set of CETs (fig. 1, 7) or
- by the Negotiated Termination set of CETs (fig. 1, 8).

Interest $cAPR$ is implicitly pre-recorded into the DLC and accounted for by appropriately changing over time: payoff amount in fig. 1, 6, collateralization in fig. 1, 7, and loan outstanding value in fig. 1, 8. As an optimization to reduce the number of CETs, interest is calculated monthly.

4.3 Loan Payoff

Payoff CET

Each CET is active for a particular time, corresponding to a specific payoff amount. The number of CETs depends on the CIG contract parameter, i.e., how frequently the acquired interest is calculated; each CET is associated with a different burn address.

Let's review the payoff sequence associated with an individual CET.

The CET encapsulates a multi-step process:

1. To release collateral, one must burn L15$(fig. 2).

   For that, in L15:

   - Alice requests burn
   - Validators check collateral state and confirm burn

2. In Bitcoin: Collateral is released after the validation of burn.

To burn L15$, there is a procedure that consists of Burn Request (fig. 2, 1) and Burn Confirm (fig. 2, 2) transactions, and an L15-specific script opcode OP_BURN_REQUEST (fig. 2, 3) to signal the beginning of the burn process.

According to the L15 consensus rules, the L15$ burn transaction is invalid unless there is an associated with it Bitcoin transaction that unlocks collateral. This is

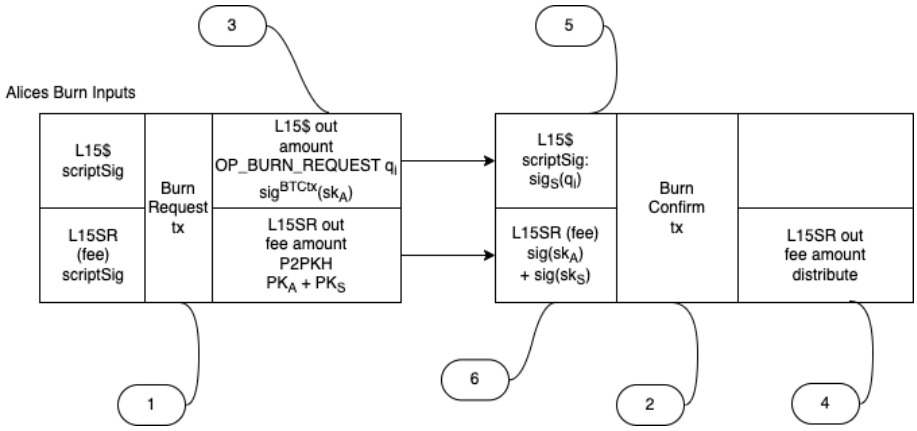

Figure 2: Burning L15$

validated twice: firstly, signers validate it by claiming corresponding fees (fig. 2, 4), and secondly, finality is ensured by miners performing PoW.

Burn in L15 Chain

To initiate the payoff, Alice creates a burn transaction using OP_BURN_REQUEST. Then, L15vroutenode reveals the secret $sig_S(q_i)$ (fig. 2, 5) formed like an oracle's broadcast for DLC:

$sig_S(q_i) = r_S - h(q_i, PK_S, R_S) \cdot sk_S$

where q_i is the burn ID that corresponds to CET_i:

$q_i = h(amount_i || R_S)$;

R_S is the ephemeral public key from L15vroutenode generated for CET_i.

The secret $sig_S(q_i)$ is used as the second private key for the multisig that locks Alice's collateral. A corresponding public key was created at contract negotiation:

$PK_{q_i} = R_S - h(q_i, PK_S, R_S) \cdot PK_S$

Bitcoin Release

Trustless Validation Loan payoff (fig. 1, 5) is an unlock of a particular UTXO encumbered in Bitcoin DLC based on a burn event in the L15 chain. L15 uses three levels of validation for a transaction like that.

Level 1 The first level is secured by L15signers that countersign the DLC. Since they are using *K-of-N* threshold multisig, to start the attack, one needs

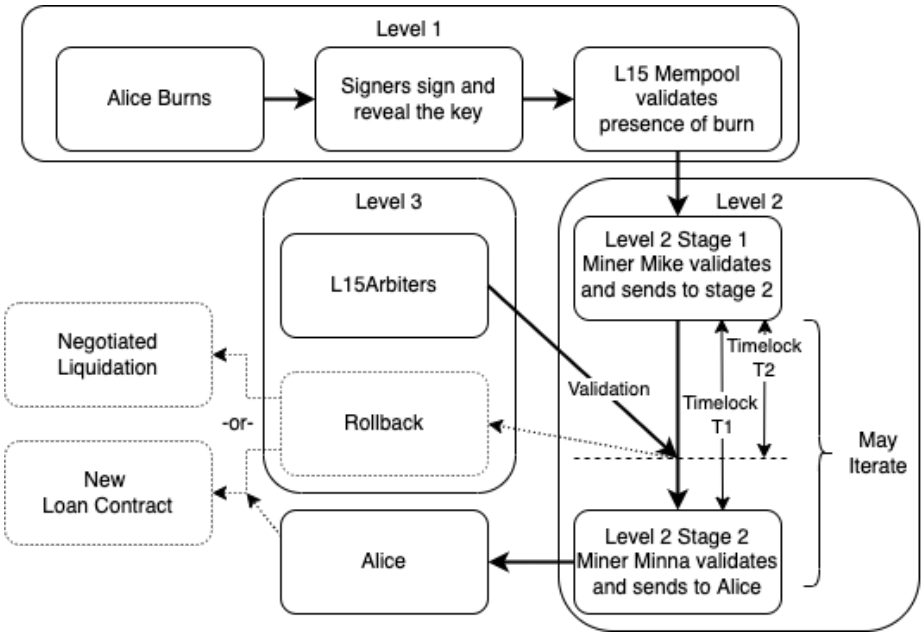

Figure 3: Three Levels of Trustless Validation

to identify *K-of-N* nodes, randomly selected for the DLC, and make them an offer worthy of losing their bond. K nodes, working together, are capable of producing the signature to release the collateral. What makes the attacker's job easier is that once selected, largely the same subset of the L15signers provide signatures for the same loan through the end of the contract lifetime.

Level 2 The second level is secured by the L15miners. L15 chain merged-mined with Bitcoin, and once in a while, an L15miner (Mike) gets to create the Bitcoin block. In order to produce an equivalent of a signoff by an L15miner, the collateral-release CET contains a one-satoshi output. Due to the Bitcoin dust limitation (546 satoshis at the time of writing), the only entity that can include a transaction with a one-satoshi output into the block is the miner who records that block, i.e., Mike. But before that, the transaction is validated by L15 nodes; the mempool will not accept an invalid payout, and Mike will not see it.

Level 2 Stage 1: According to the L15 consensus rules, Mike must validate the payout transaction before including it into the block candidate. Once Mike has mined the block with a payout transaction, the preparatory stage (stage 1) for the second level validation is complete. At that stage, the system is still vulnerable to an attack when Alice secretly obtained the unlock key from L15signers and passed it to Mike. After that, Mike can wait for as long as

necessary until he wins a block where he would include the payout transaction without a corresponding burn.

Stage 2 is designed to mitigate an attack like that.

Level 2 Stage 2: Instead of paying Alice, outputs from stage 1 are spendable either by a stage 2 validation transaction after a relative time delay T1 or by a rollback transaction after a relative time delay T2, where T1 > T2. Both T1 and T2 allow to clear potential normal reorgs in the L15 chain, but the time gap between T1 and T2 creates a possibility to rollback the payoff before collateral is sent to Alice. Stage 2 validation transaction also has a one-satoshi output, which requires another L15miner to mine another new Bitcoin block – let's call her Minna. Like Mike, Minna also must validate it before including the payout transaction into the block candidate. Unlike Mike, Minna has to use the outputs that were a public record for at least the time T1 and could have been rolled back if they were to break the L15 consensus rules. The transaction's output included in Minna's block could be a payout to Alice or it can lead to another iteration of step 2, which will strengthen the security at the expense of the increased time to release collateral.

When validating, miners can either let the transaction happen or do nothing. The latter case is covered by the third level of validation, which works in parallel with the second one, performing rollbacks if necessary.

Level 3 The third level is secured by L15arbiters. They need to act only if there is a problem discovered at level 2. Their tool to counteract the release is rollback action, which sends the collateral to Negotiated Termination. It is more pricey for Alice than the payoff but does not bear the expense of At-risk Liquidation. Naturally, in such a case, the L15 consensus rules of OP_BURN_REQUEST invalidate the burn transaction, if one exists. Negotiated Termination is actualized after a delay long enough for Alice to recreate a loan contract if she desires.

Release Pseudoscript Figure 4 adds script-level detail to the validation and release process. Compared to what was previously described, it adds "Rollback by Alice" branches that provide a failsafe exit for Alice.

Signatures in solid boxes are created at the time of contract creation, signatures in dashed boxes need to be supplied at the actualization of the respective code branch;

$l15\_clreorg\_delay + to\_self\_delay = T1$; $l15\_clreorg\_delay = T2$.

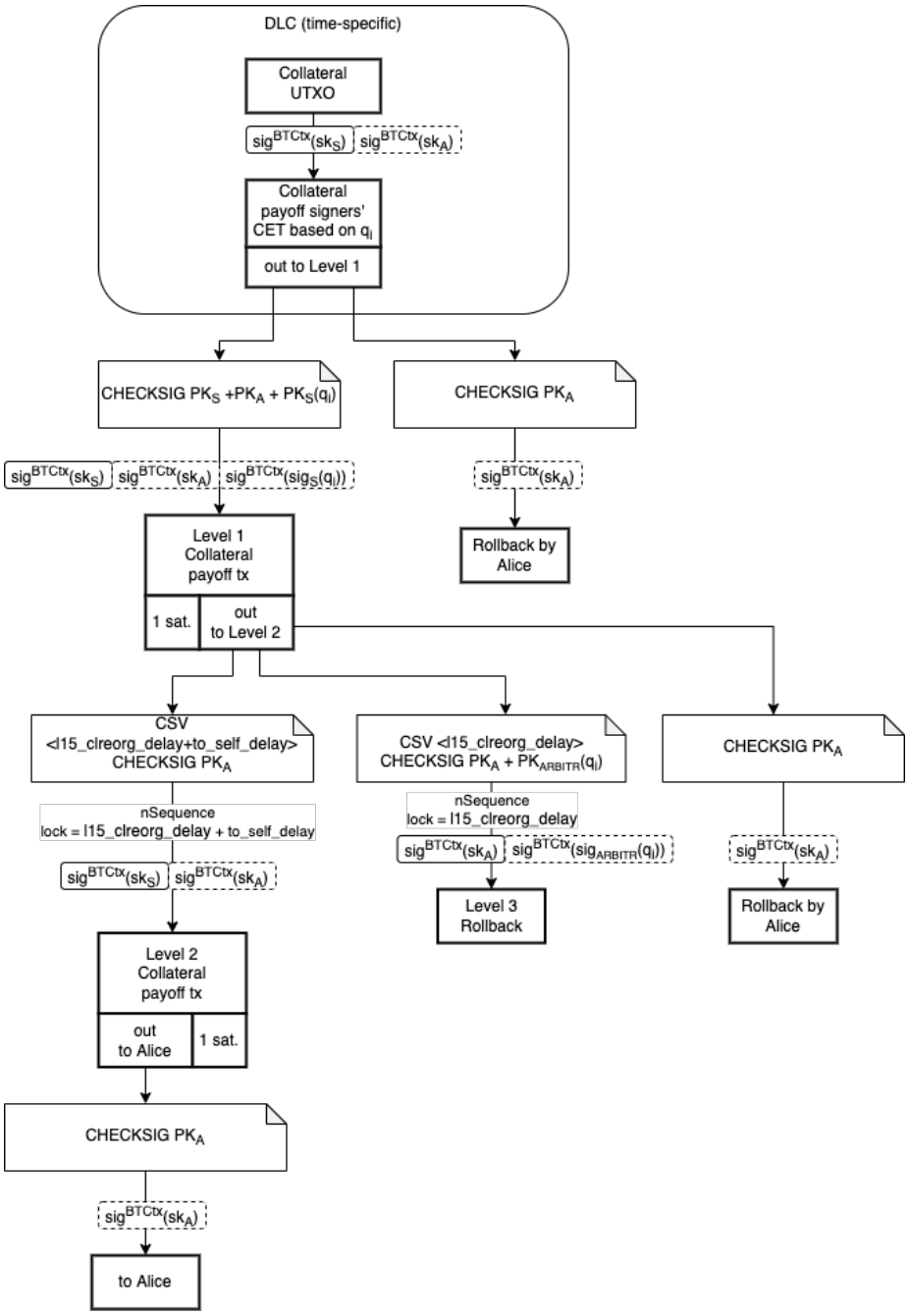

Figure 4: Bitcoin Release Pseudoscript

4.4 Contract Termination

At-risk Liquidation

When At-risk Liquidation conditions are actualized, collateral is sent to the liquidation pool.

Part of the collateral is sold for L15$ at a discounted price to cover the outstanding loan amount and fees. The remaining collateral in BTC is paid back to Alice.

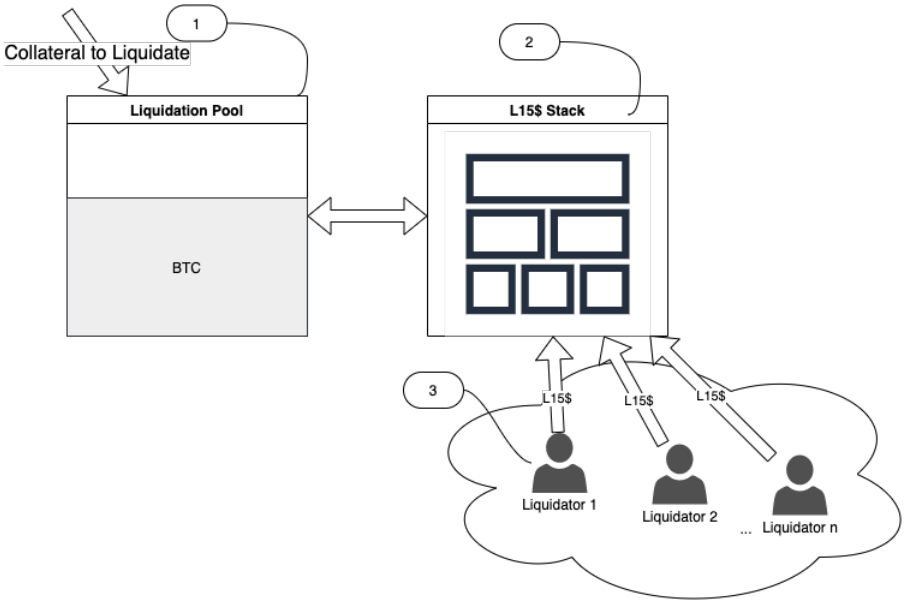

Figure 5: Liquidation

The core components for At-risk Liquidation are the Liquidation Pool (fig. 5, 1) and L15$ Stack (fig. 5, 2). The goal for L15 is to keep L15$ Stack funded at all times while the Liquidation Pool is normally empty.

Liquidators (fig. 5, 3) lock their liquidity in L15$ Stack for a predefined time, anticipating profits from future liquidations. In case of oversupply of potential Liquidators, they will be placed in the liquidation pool first. In case of excess demand in Liquidators, the system gradually changes the terms offered to Liquidators in their favor. Changing terms include interest rate, lockup duration, and liquidation reward. The liquidity is committed and locked together with the terms; Liquidator's funds are placed in a first-come-first-served queue to fund the liquidations.

While there are no liquidations, Liquidators are making interest on their locked

L15$. Once the collateral arrives in the Liquidation Pool, part of it is sold to the next liquidator in the queue at market price to get back the loaned stablecoin. Additionally, according to the terms, a liquidation reward is added to the Bitcoin sold. The latter is funded by $CTF + CLP$ and the excess overcollaterization if needed.

Negotiated Termination

Negotiated Termination occurs if the loan is not paid or re-instantiated by the end of the loan duration (CLD). Alice can also initiate it; in such a case, it most frequently is used as a fail-safe path. This option is available to Alice as long as $CL^{CURRENT} \geq CLE$. If $CLE > CL^{CURRENT} \geq CL$, Negotiated Termination is unavailable to Alice and is replaced by At-risk Liquidation. That can be corrected by Alice posting additional collateral. Additionally, Negotiated Termination has a delay for Alice to re-instantiate her loan.

How Liquidation and Negotiated Termination are Different

From Alice's standpoint, Liquidation and Negotiated Termination differ by fees and buyer's incentives. In the case of Negotiated Termination, only CTF is charged, and buyers are making regular purchases of the stablecoin at market price. In the case of Liquidation, both $CTF + CLP$ are charged. Buyers are taking a risk and expecting profits in a form of the liquidation reward.

In either case, sale proceeds must cover the outstanding loan amount and fees/rewards, and the remaining collateral is paid back to Alice. The L15$ obtained by the system at the collateral sale is burned.

4.5 Collateral Sale

In case of At-riskLiquidation the sell event is triggered trustlessly and executed via the standard means of DLC (Dryja, 2017), using Olivia's broadcasts. Upon the actualization of appropriate conditions, collateral becomes spendable by a CET(fig. 6, 5) that enables validators to send it to a process similar to Payoff, but the Buyers are the ones burning L15$ instead of Alice.

The sale transaction is the same for both At-risk Liquidation and Negotiated Termination. Note that the CET(fig. 6, 5) is one of the possible outcomes of CET sets from either At-risk Liquidation (fig. 1, 7) or Negotiated Termination (fig. 1, 8). In the case of the latter, the Oracle key check (fig. 6, 1 and 2) does not exist in the script. Olivia's signature (fig. 6, 2) is not supplied. Instead, Alice's signature (fig. 6, 3) becomes a trigger for the sale instead of Olivia's broadcast.

When the sale is triggered, CET becomes spendable by PK_s(fig. 6, 6), i.e.,

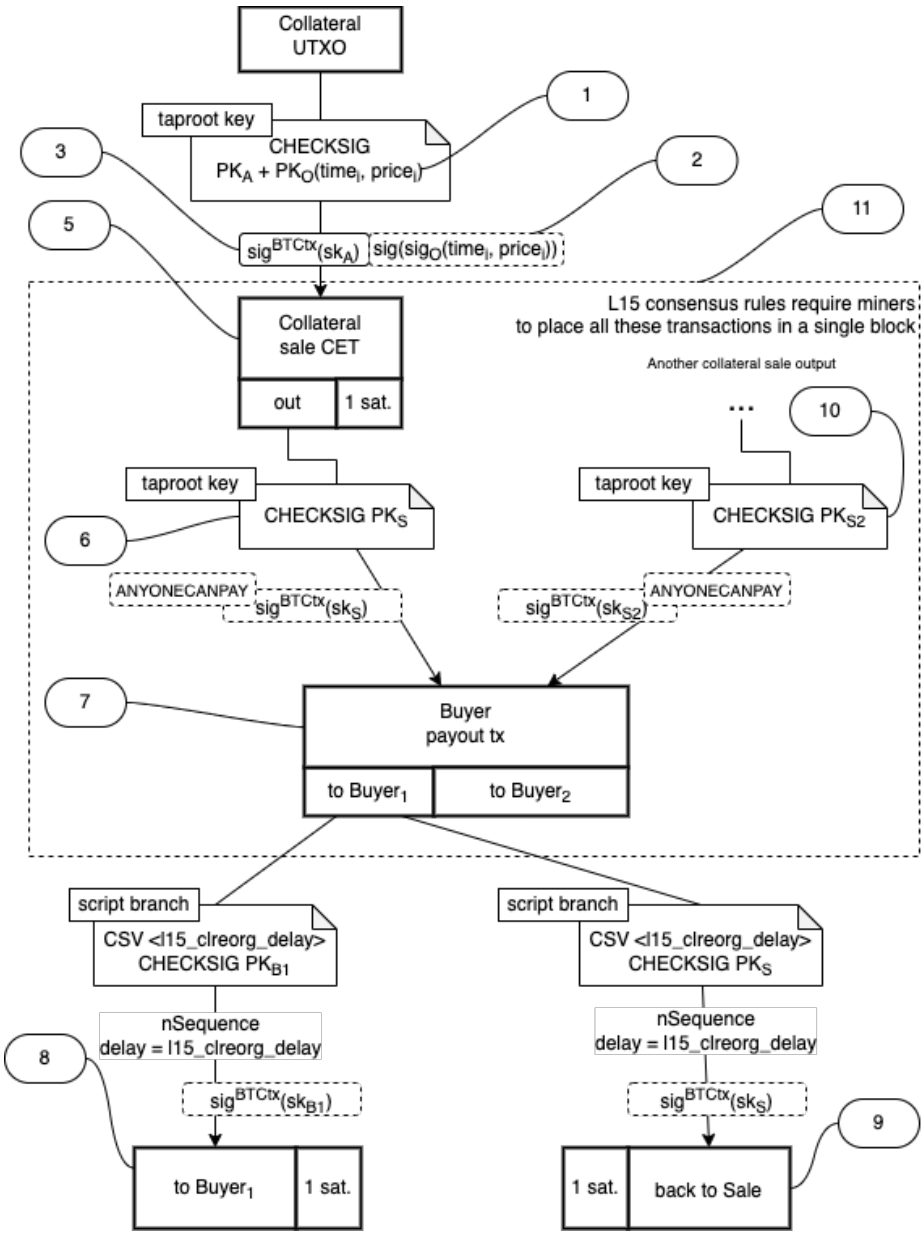

Figure 6: Collateral Sale Transaction

collateral now depends on the signature of the L15vroutenode. To use PoW validation, CET(fig. 6, 5) and spending scripts (fig. 6, 8 and 9) have a one-satoshi output. There is also a consensus rule that all transactions within the dashed box (fig. 6, 11) must be validated together and included into the same block by the same miner: Mike. At that point, payment to the buyer will be waiting for $l15\_clreorg\_delay$ and for another miner - Minna to include one of the possible branches of the transaction (fig. 6, 8 or fig. 6, 9), depending on Minna spotting any reorgs in the L15 chain which may have resulted in the rollback of buyers' burning of L15$.

Note that payout transaction (fig. 6, 7) can combine several collaterals. Figure 6 shows an additional collateral spent to the same buyer payment by a different L15vroutenode S2: fig. 6, 10.

5 State Channels

5.1 L15Lightning Interactions via the Sidechain Ledger

Let us review an example: Alice, participating in L15 as L15full_A, opens an L15lightning channel with the L15 network. Equation 2 shows the channel funding transaction as a 2 out of 2 multisig address, A, where one of the signatures is Alice's and the other is K-of-N threshold multi-sig of L15vroutnode(i) participants.

$$A = sig_{(\text{L15fullA})} + sig_{(\text{L15vroutnode(i)})} =$$

$$= \underbrace{sig_{(\text{L15fullA})} + \overbrace{(sig_{(\text{L15signer 1})} + sig_{(\text{L15signer 2})} + ... + sig_{(\text{L15signer k})})}^{\text{K-of-N}}}_{\text{2-of-2}} \quad (2)$$

Along with creating a possibility for Alice to open a channel with a "collective node", K-of-N threshold multisig creates tolerance to nodes going offline: N-K+1 signatories must be unavailable for a channel to stop working. In extreme cases, Negotiated Termination failsafe mechanism is used.

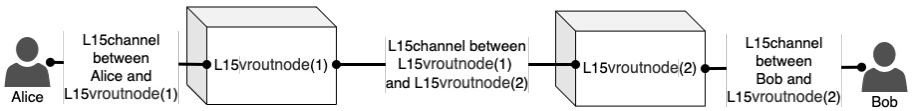

Figure 7: Alice to Bob route

Let us expand Alice's use case by adding Bob, who also has a channel with L15, and review a Lightning payment route from Alice to Bob, as shown on fig. 7. Strictly speaking, supporting a payment like this is not required for the stablecoin use case. However, we describe it as a good study example: it shows an L15 concept applied to a well-known use case of a p2p Lightning transaction.

Figure 8 further expands on the diagram fig. 7: it shows how Alice and Bob communicate with collective L15vroutenodes and transact with each other through L15channels using the L15ledger.

For simplicity of illustration, L15vroutnode(1) and L15vroutnode(2) are shown as if they consisted only of three nodes each: L15signer$_1$, L15signer$_2$ and L15signer$_3$, and L15signer$_4$, L15signer$_5$, and L15signer$_6$, respectively. Alice is shown as L15full$_\mathrm{A}$ node, and Bob is shown as L15full$_\mathrm{B}$ node.

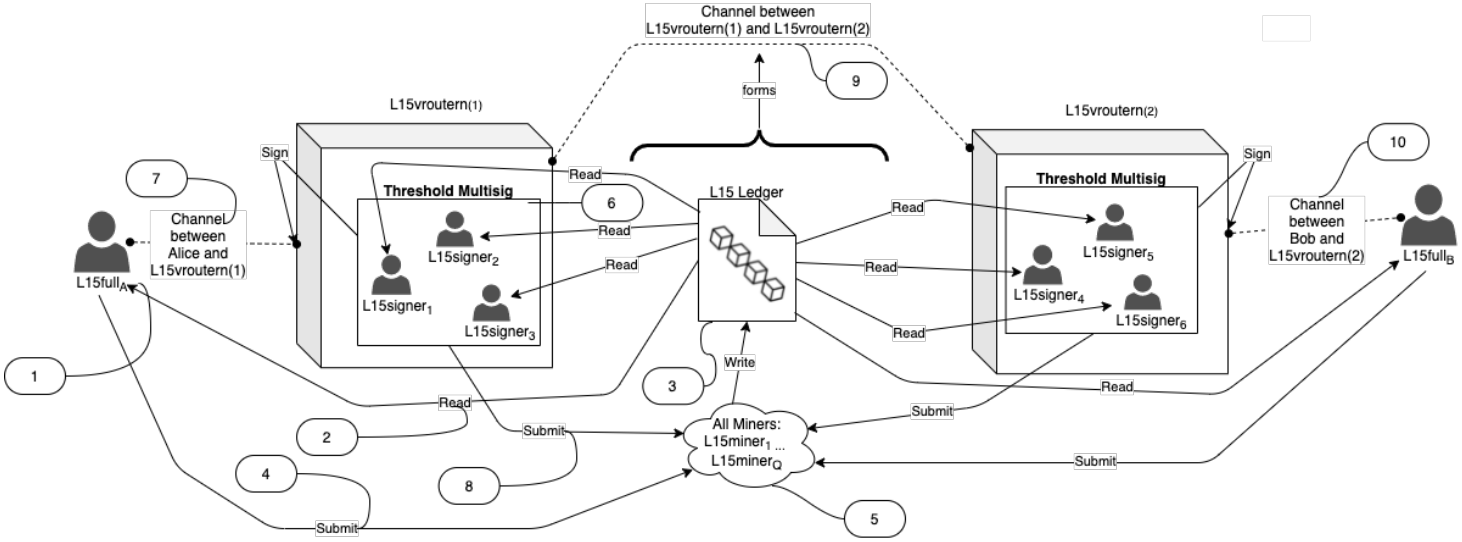

Figure 8: Alice to Bob route implemented in L15

Alice's node L15full_A (fig. 8, 1) is capable of reading (fig. 8, 2) the L15ledger (fig. 8, 3). L15full_A can submit (fig. 8, 4) records to L15 miners (fig. 8, 5) for recording in the L15ledger, Alice's L15lightning commitments are among them.

L15signer_1, L15signer_2, and L15signer_3 communicate through the L15Ledger when constructing a threshold multisig (fig. 8, 6) to sign their end of the L15channel (fig. 8, 7) with Alice. As an optimization, L15signers can communicate via mempool when building their signatures. Once the threshold signature is complete, it is submitted (fig. 8, 8) to the L15miners for recording.

L15channels(fig. 8, 7, 9, and 10) are formed and maintained by:

1. L15signers creating signatures on behalf of L15vroutnodes via K-of-N multisig,
2. L15channel participants reading the counterparties' commitments in L15ledger,
3. L15channel participants submitting newly signed commitments to the L15miners for recording in L15ledger, and
4. L15miners mining newly submitted records and returning them to all the counterparties through the L15ledger.

Compared to the Lightning Network as it exists at the time of writing, such an approach requires a change in the way commitments are formed. HTLC preimage and payment hash must be private, which is not a problem for the Lightning network since communications between its peers are direct. In the case of L15lightning, if HTLC is used, transaction data would have been made public by recording it on-chain. This data would have included payment hash and preimage, which would be unacceptable. L15 uses PTLC instead of HTLC and thus avoids using a unique preimage through a payment route to address this.

An additional challenge needs to be solved when making a p2p multihop payment via L15channels: the intermediate hops (L15vroutenodes) can not communicate their PTLC secrets privately. This challenge is not related to the use case of the stablecoin, but for completeness, we include a brief write-up about a possible solution in Appendix A.

6 BOLT Interoperability

6.1 Scope

L15 should be able to support the following:

- The payer's use of their regular Lightning Channel to fund an L15 contract; and

- The payee's use of a common Lightning channel for withdrawal of the funds from an L15 contract.

In other words, the use of L15 should not require end-users to create additional L15-specific channels. This can be fully achieved in the case of a collateral-backed loan.

While it is possible to integrate PTLC-based L15 with the HTLC-based Lightning Network, the integration would be more streamlined with PTLC available on both sides. We will base our description on the latter integration type.

6.2 Implementation

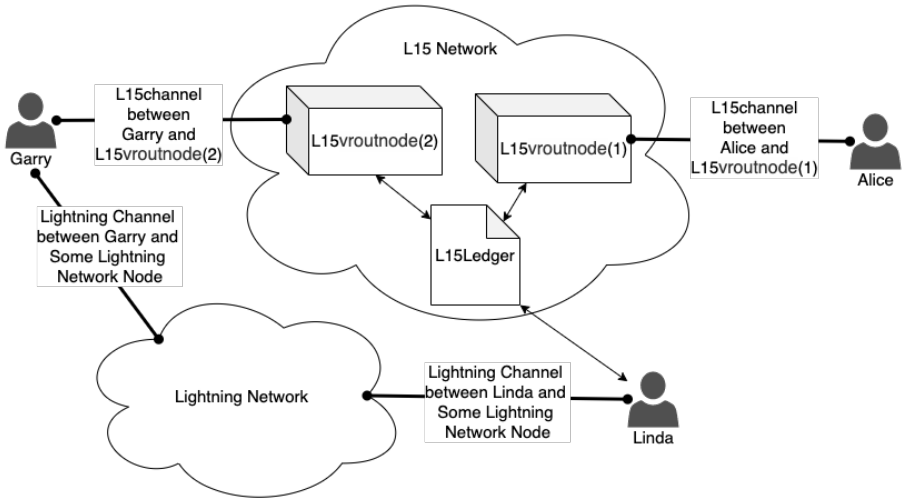

Figure 9: Linda to Alice to L15vroutenode route

Consider the network topology as shown on fig. 9. Alice, Garry, and Linda have BOLT-capable L15 clients. Linda does not have any L15channels. Alice has an L15channel with L15vroutnode(1). In our examples, we will consider Linda's case specifically: how she can access the L15 functionality utilizing only the Lightning Network channels. Linda's client exchanges data with L15 - peer-to-peer and via the L15 Ledger.

Garry has an L15channel opened with L15vroutenode(2) and one or more Lightning channels. He serves as a gateway between L15 and the Lightning Network.

Payment from L15 to a Lightning Channel

Alice sends a regular payment from her L15channel to Linda, who does not have an L15channel. This payment is made via Garry as the gateway. He reaps a benefit equivalent to "topping up" of his side of L15channel from his Lightning channel without touching Layer 1. Alice can also send funds to a Lightning Network party that is not aware of the L15 network.

This use case benefits L15 participants who desire to move their funds outside of L15channels, i.e., recieve a collateral payout.

Sending Bitcoin from a Lightning Channel to L15 DLC

Suppose Linda wants to send Bitcoin into an L15 DLC contract signed by a Ll5vroutenode. Even though Linda has no L15channels, she has to be aware of the L15 protocol:

1. Linda's client communicates with L15 through the L15ledger to create L15vroutenode(1),
2. L15vroutenode(1) provides a Lightning invoice for Linda to pay the collateral,
3. Either Linda or L15vroutenode(1):

- selects "a Garry,"
- builds a route for the payment, and
- requests PTLC lock points from the L15nodes along the route.

4. Linda sends the funds by paying the invoice. Linda must request L15vroutnodes along the payment route to generate PTLC secrets and get the lock points. This can be achieved by creating L15ledger entries that communicate with appropriate L15 participants.

7 Conclusion

L15 utilizes technologies that are becoming available in Bitcoin with the taproot upgrade: DLC, threshold multisig, and PTLC-based state channels. In addition, it uses the Bitcoin dust limit to involve its sidechain merged miners in transaction validation. As a result, L15 delivers a stablecoin soft-pegged to USD and backed by Bitcoin as collateral, built with high degree of decentralization and trustlessness.

As a vector of future development, L15 can evolve as a platform for smart contracts to enable trustless consumer finance use cases in the Bitcoin ecosystem.

Appendix A

Making a Peer-to-Peer Payment

This use case is of theoretical interest. When Alice needs to make an unconditional direct payment to Bob, it will be more efficient to use either a regular on-chain transaction or the Lightning Network.

As a starting point, let us take decorrelated payments, defined by (Malavolta et al., 2018, December 18) and detailed in (Teinturier & Jonas, 2018, November 30). L15 can not use the cited protocol verbatim because L15vroutenodes cannot either store secrets supplied from the outside, or communicate privately. Instead, L15vroutenodes generate their own secret items and communicate resulting item lockpoints via the L15ledger. Secrets to generating lockpoints are formed as a sum of secret items, one of which is unpublished and owned by the L15vroutenode that operates a given lockpoint.

L15 has to solve the challenge of sharing hashes of the upstream transactions with related downstream L15vroutenodes and preserving the privacy of transaction routes simultaneously. To address this, the route is divided into two segments - S1 and S2, each containing a set of sequential L15vroutenodes along the route; S1 is generated and managed by Alice, and S2 - by Bob; they agree upon the rendezvous node directly. Using distributed asymmetric key cryptography with threshold (Gennaro & Shoup, 2001), L15vroutenodes from S1 and S2 form their respective decryption keys and generate individual keys for Alice and Bob to encrypt the routing data about the upstream transaction hashes for S1 and S2. Alice and Bob then record the encrypted data into L15ledger. An outside observer can't connect the two segments deterministically.

References

Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J., & Wuille, P. (2014, October 22). *Enabling blockchain innovations with pegged sidechains* [White Paper]. https://blockstream.com/sidechains.pdf

Cohen, N. (2021, May 07). *Multiple oracle support* [GitHub]. https://github.com/discreetlogcontracts/dlcspecs/blob/master/MultiOracle.md

Dai (cryptocurrency). (2021, August 26). [Wikipedia Article]. https://en.wikipedia.org/wiki/Dai_(cryptocurrency)

Dryja, T. (2017). *Discreet log contracts* [White Paper]. https://adiabat.github.io/dlc.pdf

Gennaro, R., & Shoup, V. (2001). *Securing threshold cryptosystems against chosen ciphertext attack*. https://www.shoup.net/papers/thresh1.pdf

Guillyr, T. L., Benthecarman, & Cohen, N. (2020, November 03). *Oracle specifications* [GitHub]. https://github.com/discreetlogcontracts/dlcspecs/blob/master/Oracle.md

Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., & Maffei, M. (2018, December 18). *Anonymous multi-hop locks for blockchain scalability and interoperability* [White Paper]. https://eprint.iacr.org/2018/472.pdf

Poelstra, A. (2015, March 22). *On stake and consensus* [White Paper]. https://nakamotoinstitute.org/static/docs/on-stake-and-consensus.pdf

Teinturier, B., & Jonas, N. (2018, November 30). *Multi-hop locks from scriptless scripts* [GitHub]. https://github.com/ElementsProject/scriptless-scripts/blob/master/md/multi-hop-locks.md