Flyover: A Repayment Protocol for Fast Bitcoin Transfers over Federated Pegs

Javier Álvarez Cid-Fuentes[0000−0001−7153−4649], Diego Angel Masini, and Sergio Demian Lerner

IOV Labs Ltd.
{javier, dmasini, sergio}@iovlabs.org

Abstract. As the number of blockchain projects grows, efficient cross-chain interoperability becomes more necessary. A common cross-chain protocol is the two-way peg, which is typically used to transfer assets between blockchains and their sidechains. The criticality of cross-chain protocols require that they are designed with strong security models, which can reduce usability in the form of long transfer times. In this paper, we present Flyover, a repayment protocol to speed up the transfer of bitcoins over federated pegs by allowing untrusted liquidity providers to advance funds for the users. Transfer times are reduced because liquidity providers do not have the same security requirements as the underlying cross-chain protocol. We illustrate the Flyover protocol on the cross-chain interoperability protocol that connects Bitcoin to the RSK sidechain and show how Flyover can reduce transfer times without reducing security. In addition to this, Flyover extends the cross-chain protocol by allowing liquidity providers to make smart contract calls on RSK on behalf of the user.

1 Introduction

The applicability of blockchain technology in different domains is growing at a fast pace, creating a diverse number of use cases and a heterogeneous ecosystem of blockchains [4]. At the same time, innovative technologies, such as sidechains [14], networks of blockchains [16], and off-chain protocols [7] are emerging as a solution to various limitations of current blockchains. Including scalability, decentralization, usability, and performance limitations. This creates a fragmented ecosystem that makes cross-chain interoperability crucial [4, 13].

Two-way pegs [3] are one of the existing cross-chain interoperability protocols typically used to transfer assets between blockchains and their sidechains [14]. Two-way pegs are a critical component of sidechains, as these cannot operate without a cross-chain communication protocol that connects them to their parent chain [5, 6, 8]. As a critical component, two-way pegs must be designed with a strong security model, which might limit their usability [14].

An example of a sidechain that uses a federated two-way peg is Rootstock (RSK) [8], which provides stateful smart contract functionality on top of Bitcoin [11]. The two-way peg that connects RSK to Bitcoin is called the Powpeg.

One drawback of the strong security requirements of the Powpeg is that transfers from Bitcoin to RSK require 100 Bitcoin block confirmations and transfers from RSK to Bitcoin require 4,000 RSK block confirmations. This translates to around 16 and 33 hours respectively. These transfer times increase the security of the Powpeg in case of blockchain reorganizations, but also limit the usability of the RSK sidechain.

In this paper, we present Flyover, a novel protocol to transfer BTC between Bitcoin and RSK in a faster way than using the Powpeg. Flyover is a repayment protocol that allows liquidity providers to advance funds for the user. After the transfer to the user is completed, liquidity providers are repaid the advanced funds plus a service fee. In this way, users can potentially receive transfers much faster as liquidity providers have lower security requirements than the Powpeg. In addition to this, Flyover allows liquidity providers to call smart contracts in RSK on behalf of the user. This enables smart contract calls in RSK directly from Bitcoin.

Flyover provides faster BTC transfers without reducing security by relying on the Powpeg. In the worst case scenario, the process either falls back to the the regular Powpeg mechanism or the protocol allows the user to cancel the transfer. Liquidity providers never have direct access to the user funds, and thus, users are never at risk of losing their money. In addition, the Flyover protocol also implements punishing mechanisms to prevent liquidity providers from misbehaving.

The remainder of this document is organized as follows: Section 2 overviews repayment protocols similar to Flyover; Section 3 presents the background knowledge needed to fully understand the Flyover protocol; Section 4 describes the Flyover protocol in detail; and Section 6 presents our conclusions.

2 Related work

The Powpeg implements a common cross-chain protocol to transfer assets between blockchains that has been formalized by XLCAIM [17] more recently. The general idea of the protocol is that transferring an asset C_a from a blockchain B_a to a blockchain B_b consists of locking C_a in a vault in B_a and then minting an equivalent quantity of an asset C_b on B_b. The inverse transfer consists of burning C_b in B_b to unlock the equivalent quantity of C_a in B_a. There are several projects that implement this type of cross-chain protocol, including Polkadot [16], pTokens [2], and Liquid [12]. Among these, Liquid uses a federated peg similar to the Powpeg that connects Bitcoin to their sidechain. Flyover improves these types of protocols through a repayment scheme. That is, Flyover introduces an untrusted third party that advances the funds for the user and later is repaid. This allows Flyover to reduce transfer times without reducing the security of the process. Two repayment protocols similar to Flyover are MakerDAO's Optimism DAI bridge [9] and Hop [15].

MakerDAO's DAI bridge uses a loan system to speed-up DAI token transfers from the layer-2 solution Optimism [1] to the underlying blockchain. These transfers take usually one week due to security reasons. The repayment protocol allows users to get a collateralized loan before the transfer is fully verified by the bridge. The loan is issued directly by MakerDAO's treasury to guarantee the consistency of the bridge even in case of misbehavior. After the one week period, users can repay the loan and recover the DAI tokens from the bridge. The main difference between MakerDAO's DAI bridge and Flyover is that in Flyover users receive the funds directly instead of a loan. This makes MakerDAO's protocol more complex, as users must be incentivized to repay the loan after one week. In addition to this, MakerDAO's bridge relies on an oracle to get information from the layer-2, which represents a centralized point of failure that does not exist in Flyover.

Hop is a repayment protocol to speed-up token transfers between rollups [10] that removes the need to go through the underlying blockchain. The idea is to use a special Hop token that can be burned at the source rollup and minted at the destination. To achieve this, a *Bonder* provides the liquidity for the user at the destination rollup. The Bonder is then refunded (plus a fee) when the payment propagates to the underlying blockchain. Bonders can check that the transfer at the source rollup is valid by running a verifier node for the rollup. In addition, the Hop protocol also establishes automated market makers to convert Hop tokens to the canonical token used in each rollup. The main difference between Hop and Flyover is that Flyover is built on top of a federated two-way peg, while Hop connects layer-2 solutions built on a shared blockchain.

3 Background

Rootstock (RSK) [8] is a Bitcoin sidechain that provides stateful and turing-complete smart contract functionality. Although RSK implements its own consensus mechanism, it does not require a money token different from bitcoin (BTC) to function. That is, fees and computations in RSK are paid in BTC. This prevents RSK from competing against Bitcoin as a store of value and aligns RSK's incentives to Bitcoin. RSK provides a highly secure bridging protocol, called the Powpeg, to transfer bitcoins from the Bitcoin network and back. For clarity, we will refer to bitcoins in RSK as RBTC.

The fact that RSK relies on BTC to operate makes the Powpeg an essential part of the RSK network. The Powpeg is a two-way peg that uses a vault in Bitcoin, where users can deposit BTC, and a smart contract in RSK that locks and unlocks RBTC as requested. Consistency and liveness in the Powpeg are guaranteed by means of three components: a set of Pegnatories, a set of Hardware Security Modules (PowHSMs), and a pre-compiled Bridge smart contract in RSK.

Pegnatories run a modified RSK node and act mainly as data relays between the other components of the Powpeg and the Bitcoin network. The PowHSMs hold the private keys that control the Bitcoin vault, which consists of a multi-signature script. Each Pegnatory maintains one PowHSM but does not have direct access to the corresponding private key. The pre-compiled Bridge contract serves as a Bitcoin SPV client in RSK.

Figure 1 shows the *peg-in* process, that is, the process of transferring bitcoin from Bitcoin to RSK. The process consists of the following steps:

1. The user sends N BTC to the multi-signature address controlled by the PowHSMs. This is the deposit transaction.
2. The Pegnatories, or other users, periodically update the Bridge contract with Bitcoin block headers. The Bridge contract selects the best chain based on cumulative work.
3. The Pegnatories, or the user, submit the deposit transaction to the Bridge smart contract on RSK.
4. The Bridge contract validates the deposit transaction using its internal view of the Bitcoin blockchain.
5. After 100 block confirmations, the Bridge contract transfers N RBTC to the user in RSK. The destination address in RSK is derived from the source address in Bitcoin.

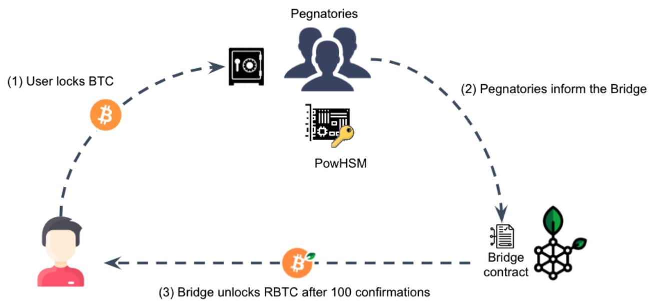
Pegnatories
(1) User locks BTC
(2) Pegnatories inform the Bridge
PowHSM
Bridge contract
(3) Bridge unlocks RBTC after 100 confirmations

Fig. 1: Powpeg's peg-in process.

The Bridge contract requires 100 block confirmations to release the RBTC to guarantee the security of the Powpeg in the event of Bitcoin chain reorganizations. In addition to this, the Powpeg establishes a limit in the amount of BTC that is locked in the vault. If the locked amount exceeds this limit due to a user deposit, the Powpeg immediately refunds the user in BTC instead of performing the transfer to RSK.

The *peg-out* process, that is, the process of transferring bitcoin from RSK to Bitcoin is more complex because it involves unlocking BTC from the vault in a secure manner. Figure 2 shows a diagram of this process, which consists of the following steps:

1. The user sends N RBTC to the Bridge contract.
2. After 4,000 confirmations, the Bridge contract builds a Bitcoin transaction that unlocks BTC from the vault.

3. The Pegnatories collect the Bitcoin transaction and send it to their PowHSMs.
4. The PowHSMs verify that the Bitcoin transaction has been generated by the Bridge in an RSK block with a specific amount of proof-of-work (PoW) on top.
5. After validation, the PowHSMs sign the transaction and the Pegnatories send the signatures to the Bridge contract.
6. The Bridge puts together the signed transaction when it receives more than half of the PowHSM signatures.
7. The Pegnatories relay the signed transaction to the Bitcoin network and the user receives N BTC from the vault.

As with the peg-in process, the Bridge contract requires 4,000 confirmations due to security reasons. In addition to this, Pegnatories do not directly control the locked BTC. Instead, the PowHSMs require a particular amount of PoW to sign a BTC unlocking transaction. This makes stealing from the vault economically unfeasible as Pegnatories would need to spend computational resources in mining a significant amount of fake RSK block headers.

4 The Flyover protocol

The Flyover protocol introduces an untrusted third party, called the Liquidity Provider (LP), that advances bitcoin for the user. In this way, the protocol can potentially achieve faster transfers than the Powpeg. This is because LPs might require less confirmations, especially for small value transfers, as their security requirements are less strict than the Powpeg's. A breach in the consistency of the Powpeg would be fatal for the RSK network, while LPs can operate with higher risk assumptions.

The Flyover protocol requires changes to the pre-compiled Bridge contract, a new smart contract in RSK called the Liquidity Bridge Contract (LBC), and the intervention of an LP. In terms of RSK consensus, the Flyover protocol only defines the interactions with the Bridge contract. The internals of the LBC and the LP can be defined in several ways, and multiple third-parties can deploy

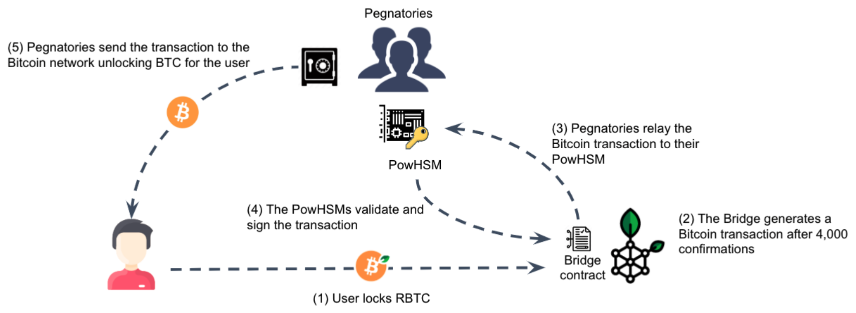
Pegnatories
(5) Pegnatories send the transaction to the Bitcoin network unlocking BTC for the user
(3) Pegnatories relay the Bitcoin transaction to their PowHSM
PowHSM
(4) The PowHSMs validate and sign the transaction
(2) The Bridge generates a Bitcoin transaction after 4,000 confirmations
Bridge contract
(1) User locks RBTC

Fig. 2: Powpeg's peg-out process.

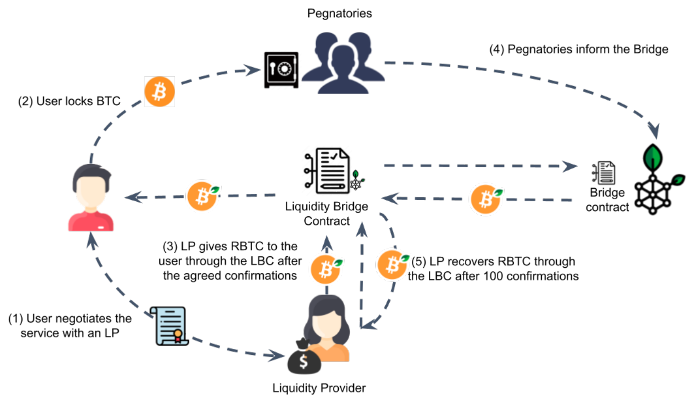
Pegnatories
(4) Pegnatories inform the Bridge
(2) User locks BTC
Liquidity Bridge Contract
Bridge contract
(3) LP gives RBTC to the user through the LBC after the agreed confirmations
(5) LP recovers RBTC through the LBC after 100 confirmations
(1) User negotiates the service with an LP
Liquidity Provider

Fig. 3: Overview of the Flyover peg-in process.

their own components. This creates a potential marketplace of fast bridges that compete against each other. In the following, we describe the peg-in and peg-out processes in detail.

4.1 Peg-in process

A peg-in using the Flyover protocol begins with an off-chain interaction between the user and the LP in which they negotiate the conditions of the service. The user then makes a deposit in BTC to a special address that is controlled by the same private keys that control the multi-signature address used in the Powpeg. The LP verifies the deposit transaction and pays the user the agreed amount in RBTC through the LBC. Finally, after the usual 100 block confirmations, the Bridge contract pays the LBC as in a regular Powpeg peg-in and the LBC refunds the LP.

Figure 3 shows a diagram of the main interactions in Flyover during the transfer of BTC to RSK. Note that the protocol makes use of the Pegnatories as data relays between Bitcoin and the Bridge contract, but users can also relay the information themselves.

The peg-in process requires the addition of a new method to the pre-compiled Bridge contract called `registerFastBTCTransaction` that takes the following arguments:

`btcTransaction` – The user deposit transaction that locks BTC.
`height` – The height of the block that contains the deposit transaction.
`merklePath` – The inclusion proof of the deposit transaction in the block.
`derivationHash` – A hash used as unique identifier for the peg-in.
`btcRefundAddress` – The user Bitcoin refund address.
`lbcAddress` – The LBC RSK address.
`lpRefundAddress` – The LP Bitcoin refund address.
`callPerformed` – A boolean indicating whether the LP provided the service.

The LBC uses the `registerFastBTCTransaction` method to validate the user BTC deposit and to get the corresponding amount of RBTC after the 100 block confirmations. Apart from this, the LBC makes use of a Bridge method called `getBtcBlockchainBlockHeaderByHeight`, which returns the Bitcoin block header at a given height. We explain the details on how the LBC uses these two Bridge methods below.

Off-chain interaction The first step in transferring BTC to RSK consists of an off-chain negotiation of the service conditions between the user and the LP. In this interaction, the user and the LP agree on a service quote. The LP then commits to delivering the service by signing the hashed quote. The signed hashed quote can be used later to punish the LP in case of misbehavior. For this, LPs are required to lock collateral with the LBC. Peg-in quotes consist of the following fields:

`PegBitcoinAddress:` (20 bytes) the Powpeg Bitcoin multi-signature address.

`LBCAddress:` (20 bytes) RSK address of the LBC.

`LPRSKAddress:` (20 bytes) RSK address of the LP.

`BitcoinRefundAddress:` (21 bytes) user Bitcoin refund address (including version prefix).

`RSKRefundAddress:` (20 bytes) user RSK refund address.

`LPBitcoinAddress:` (21 bytes) Bitcoin address of the LP (including version prefix).

`CallFee:` (8 bytes) unsigned integer representing the fee that the LP charges.

`PenaltyFee:` (8 bytes) unisgned integer representing the penalty applied to the LP in case of misbehavior.

`DestinationAddress:` (20 bytes) RSK address that receives the transfer. Can be an externally owned account or a contract address.

`Data:` byte array containing the call arguments in case of a smart contract call.

`GasLimit:` (4 bytes) unsigned integer representing the gas limit used in the call.

`Nonce:` (8 bytes) integer that uniquely identifies this quote.

`Value:` (8 bytes) unsigned integer representing the value to transfer.

`Timestamp:` (4 bytes) unsigned integer representing the time of the agreement.

`TimeForDeposit:` (4 bytes) unsigned integer representing the time that the user has to make the deposit transaction.

`Confirmations:` (2 bytes) number of confirmations that the LP requires before delivering the service.

`CallTime:` (4 bytes) unsigned integer representing the time that the LP has to deliver the service after the required number of confirmations have occurred.

Some of these fields define the conditions of the service, while others serve to uniquely identify the LP, the LBC, and the quote itself. During the off-chain negotiation, the user informs the LP about the value to be transferred, the destination address, and the data and gas limit of the call. The LP may then choose the appropriate fees, number of required confirmations and other quote fields based on this information.

Deposit transaction After receiving the signed quote, the user must make a deposit to a special Bitcoin address meeting two conditions: first, the deposit must be greater or equal to `Value` plus `CallFee`; second, the deposit must achieve one block confirmation before `Timestamp` plus `TimeForDeposit`. If the deposit does not meet one of these conditions, the LP is not compelled to deliver the service and the user receives their funds after the usual 100 block confirmations. The LP and the user are responsible for choosing the appropriate values for these fields taking into account the status of the Bitcoin and RSK networks.

The deposit address is derived from the multi-signature address controlled by the Powpeg. This multi-signature address is a pay-to-script hash (P2SH) address that can be unlocked by providing the corresponding script and the number of required signatures. The Flyover deposit address is generated by hashing together the quote hash and the `BitcoinRefundAddress`, `LBCAddress` and `LPBitcoinAddress` quote fields. The resulting hash is placed at the beginning of the multi-signature locking script followed by the `OP_DROP` instruction. This instruction discards the preceding value, and thus, the behavior of the resulting locking script remains the same. In this manner, the Flyover deposit address is linked to a specific quote, and to specific user, LP, and LBC addresses, while still being controlled by the same private keys that control the Powpeg P2SH address. This ensures that the Powpeg can unlock the BTC when users request a transfer from RSK to Bitcoin, and also prevents claiming a deposit using invalid parameters.

Advancement of funds The LP advances the funds to the user through an LBC function named `callForUser` that takes the quote as argument. The LP calls `callForUser` after the user deposit achieves the number of block confirmations specified in the `Confirmations` quote field. The `callForUser` function then makes a call to `DestinationAddress` transferring an amount of RBTC equal to `Value`; passing `Data` as input data for the call; and setting the gas limit of the call to `GasLimit`. This is possible because, like in Ethereum, a transaction in RSK can be a transfer of value, a contract call, or a contract call that also transfers value. The `Data` field of the quote allows users to directly call smart contracts in RSK from Bitcoin through the Flyover protocol in a seamless manner.

The LBC checks that the `LPRSKAddress` quote field corresponds to the caller address, that the LP has enough liquidity to perform the call, and that the quote has not been processed already. The LBC also stores the time at which the LP calls the `callForUser` function. Note that it is the LP's responsibility to ensure that `CallFee` is enough to cover for the costs of interacting with the LBC.

Resolution The peg-in process can be finalized after 100 block confirmations on the user deposit. There are several ways in which the process can resolve. In all cases, resolution is through an LBC function called `registerPegIn`. The `registerPegIn` function can be called by the LP, the user, or any other entity, such as another LP. The function takes five arguments: the quote, the quote signature, the serialized Bitcoin deposit transaction, a Merkle tree path proving the inclusion of the transaction in the block, and the height of the Bitcoin block that contains the deposit transaction. The `registerPegIn` function validates the quote by checking that the provided signature matches the quote hash and the `LPRSKAddress` quote field. Then, the `registerPegIn` function calls the `registerFastBTCTransaction` Bridge method providing the Bitcoin transaction, the height, the Merkle tree path, the quote hash, the LBC and refund addresses, and whether the LP called `callForUser`. The Bridge then validates that:

- The caller address is the provided LBC address.
- The provided quote hash and serialized Bitcoin transaction have not been processed before.
- The provided block height is at least 100 blocks from the current main blockchain tip.
- The Merkle root obtained from the provided Merkle tree path and serialized Bitcoin transaction corresponds to the Merkle root of the block at the provided height.
- The provided Bitcoin transaction contains an output to the deposit address. For this, the Bridge contract generates the deposit address from the provided quote hash and the user, LP, and LBC addresses.

As mentioned before, the Bridge contract is able to perform these validations because the Pegnatories periodically relay Bitcoin block headers. After the validations, the Bridge contract transfers the deposited amount in RBTC to the LBC. The LBC can then finalize the process in several ways:

- The LBC refunds the LP if the LP called `callForUser` before `CallTime` seconds after the deposit transaction reached the number of agreed confirmations. The LBC uses the Bridge method `getBtcBlockchainBlockHeaderByHeight` to obtain the timestamp of the Bitcoin block.
- The LBC refunds the LP but slashes its collateral by `PenaltyFee` if the LP did not call `callForUser` on time. The LBC burns part of `PenaltyFee` and pays the rest to the caller of `registerPegIn`.
- The LBC punishes the LP as before and refunds the user at `RSKRefundAddress` if the LP did not call `callForUser` at all.

In the worst case (i.e., the LP does not deliver), the user receives the RBTC after the regular 100 block confirmations by calling `registerPegIn`. Anyone can call `registerPegIn` with the incentive of receiving part of the penalty fee. This guarantees that users are never at risk of losing funds.

In the case that the user deposit makes the total amount of locked BTC in the Powpeg greater than the maximum allowed amount, the Bridge contract does not transfer RBTC to the LBC. Instead, the Bridge contract returns the BTC to `BitcoinRefundAddress` or `LPBitcoinAddress` depending on whether the LP already advanced the RBTC for the user.

4.2 Peg-out process

The peg-out process is less complex than the peg-in because it can take advantage of RSK's smart contract functionality to lock the RBTC. Similar to the peg-in, the peg-out process begins with an off-chain interaction between the user and the LP. As part of this interaction, the user and the LP compute a derived Bitcoin address controlled by the user. The LP provides a signed quote and the user makes an RBTC deposit on the LBC. After a number of confirmations on this deposit, the LP pays the agreed amount of BTC to the derived user address in Bitcoin. After this payment achieves a number of confirmations, the LP provides the LBC with a proof and gets refunded in RBTC. Figure 4 shows a diagram of the peg-out process.

Apart from `getBtcBlockchainBlockHeaderByHeight`, the peg-out process makes use another Bridge method called `getBtcTransactionConfirmations` that returns the number of confirmations for a given Bitcoin transaction. We describe the details of the peg-out process in the following.

Off-chain interaction As with the peg-in, the first step of the peg-out process is an off-chain negotiation between the user and the LP. In this case, the quote contains the following fields:

`DerivationAddress:` (21 bytes) Bitcoin derived address controlled by the user.

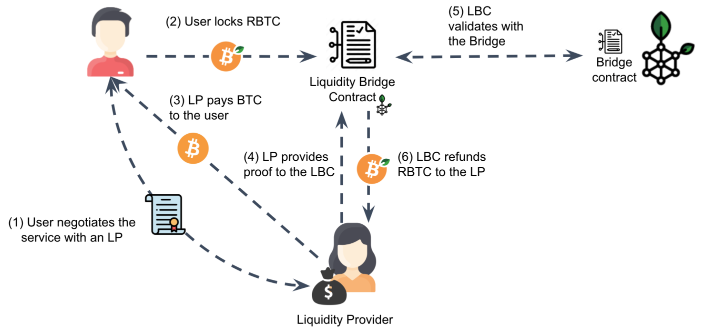
(2) User locks RBTC
(5) LBC validates with the Bridge
Liquidity Bridge Contract
Bridge contract
(3) LP pays BTC to the user
(4) LP provides proof to the LBC
(6) LBC refunds RBTC to the LP
(1) User negotiates the service with an LP
Liquidity Provider

Fig. 4: Overview of the Flyover peg-out process.

`LBCAddress:` (20 bytes) RSK address of the LBC.
`LPRSKAddress:` (20 bytes) RSK address of the LP.
`RSKRefundAddress:` (20 bytes) user RSK refund address.
`Fee:` (8 bytes) unsigned integer representing the fee that the LP charges.
`PenaltyFee:` (8 bytes) unisgned integer representing the penalty applied to the LP in case of misbehavior.
`Nonce:` (8 bytes) integer that uniquely identifies this quote.
`Value:` (8 bytes) unsigned integer representing the value to transfer.
`Timestamp:` (4 bytes) unsigned integer representing the time of the agreement.
`DepositTimeLimit:` (4 bytes) unsigned integer representing the time limit that the user has to include the RBTC deposit in the RSK blockchain.
`DepositConfirmations:` (2 bytes) number of confirmations that the LP requires before delivering the service.
`PaymentConfirmations:` (2 bytes) number of confirmations that the LBC requires on the payment transaction to refund the LP.
`TransferTime:` (4 bytes) unsigned integer representing the time that the LP has to deliver the service.
`ExpiryTime:` (4 bytes) unsigned integer representing the time after which the user can cancel the service.
`ExpiryBlocks:` (4 bytes) unsigned integer representing the number of RSK blocks after which the user can cancel the service.

As part of the negotiation, the user must provide his Bitcoin public key hash. The LP and the user then compute the derivation address as a P2SH address similar to the deposit address of the peg-in process. In this case, the locking script begins with the quote hash (without the derivation address) followed by the `OP_DROP` instruction and can be unlocked with the user public key. The resulting derivation address is controlled by the user while being linked to a specific quote. This makes the LP payment to this address unique and prevents LPs from claiming multiple user deposits with the same payment transaction.

Deposit transaction After the negotiation, the user sends the signed quote together with an RBTC deposit of `Value` plus `Fee` to the LBC by calling a method named `registerPegOut`. This deposit transaction must be included in an RSK block before `DepositTimeLimit` to be accepted by the LBC. This is to ensure that a delay in the deposit does not affect the LP's ability to deliver the service on time. The LBC stores the quote hash, the timestamp and the block number of the deposit for future validations.

Advancement of funds Once the deposit transaction gets `DepositConfirmations` confirmations, the LP pays `Value` BTC to the derivation address controlled by the user. The LP must ensure that this payment transaction is included in the Bitcoin blockchain before `TransferTime` starting at the time of the deposit transaction.

Resolution After the payment transaction gets `PaymentConfirmations` confirmations, the LP can submit a proof of the payment to the LBC to get a refund in RBTC. More precisely, the LP calls a function named `refundPegOut` with five arguments: the quote, the raw Bitcoin payment transaction, the hash of the Bitcoin block header in which the payment transaction is included, a Merkle tree path proving the inclusion of the transaction in the block, and the height of the Bitcoin block that contains the payment transaction. The `refundPegOut` method validates that:

- The quote is valid and corresponds to a pending peg-out.
- The provided Bitcoin transaction has `PaymentConfirmations` confirmations. For this, the LBC calls the Bridge method `getBtcTransactionConfirmations`.
- The Bitcoin transaction contains an output paying `Value` to the derivation address.
- The Bitcoin transaction was included in the blockchain before the time of the user deposit plus `TransferTime`. For this, the LBC uses the Bridge method `getBtcBlockchainBlockHeaderByHeight`.

After these validations, the LBC refunds `Value` plus `Fee` RBTC to the LP. The LBC slashes `PenaltyFee` RBTC from the LP's collateral if the payment was not made on time.

The user can call an LBC method named `cancelPegOut` to get a full refund and penalize the LP if the LP does not submit proof of the payment before `ExpiryTime` or `ExpiryBlocks` since the user deposit. We use two deadlines to account for network delays. This mechanism protects the user in case the LP does not deliver the service and forces LPs to claim refunds before a particular deadline.

5 Security considerations

In this section, we discuss various aspects of the security of the Flyover protocol: the use of derivation addresses, the LP risks, and the possibility of denial-of-service attacks.

5.1 Derivation addresses

The central part of the security of the Flyover peg-in process lies in the derivation of the deposit address. The Bridge only pays RBTC to the LBC address associated with the BTC deposit (or one of the refund addresses if the locking limit is exceeded). Including the quote hash in the deposit address prevents quote re-submission or claiming the RBTC using an incorrect quote. At the same time, the deposit address is controlled by the PowHSMs, which provide the same security guarantees as the Powpeg. That is, Flyover reduces the time to peg-in without reducing its safety. Users are never at risk of losing funds because anyone can trigger the resolution of the peg-in through the LBC, and LPs are also guaranteed to receive refunds through the LBC as long as they provide the service.

The derivation address during the peg-out process guarantees that each BTC payment to the user is unique. This has two main security implications. On the one hand, it prevents LPs from claiming multiple refunds using the same payment transaction. On the other hand, it also prevents LPs from claiming a refund using a different payment transaction to the same user address.

5.2 Penalty risk

During the peg-in process, the LP must call `callForUser` after the number of agreed confirmations on the user deposit but before `CallTime`. This puts the LP at risk of being penalized if there are network delays in RSK. LPs must take into account this risk when negotiating the conditions of the service and set `CallTime` and the service fee accordingly.

During the peg-out, the LP also takes the risk of getting the payment Bitcoin transaction mined before `TransferTime` after the user deposit. In this case, `TransferTime` includes the time that it takes for the user deposit to reach the number of agreed confirmations. This means that during peg-outs, LPs must take into account possible network delays both in Bitcoin and RSK.

5.3 Denial-of-service attacks

Both during the peg-in and the peg-out, the LP commits to delivering the service by signing the agreed quote. This signature serves as a proof that users can present to the LBC to penalize the LP in case of not delivering the service. This means that LPs are forced to lock part of their liquidity after signing each quote, however, users are not required to make the deposit and can let the quote expire. A malicious user could request many quotes from an LP and never make any deposits. This would force the LP to lock all of its liquidity and make it unable to serve honest users.

6 Conclusions

In this paper, we present Flyover, a repayment protocol to speed-up bitcoin transfers on federated pegs. Flyover is built on top of the Powpeg, which is the current bridging protocol between Bitcoin and RSK. Flyover achieves faster transfers than the Powpeg by having untrusted liquidity providers to advance funds for the user. This is because liquidity providers can operate under higher risk assumptions than the Powpeg. In addition, Flyover allows users to trigger smart contract calls on RSK directly from Bitcoin.

Although liquidity providers are external untrusted parties, the security of Flyover is guaranteed by the same mechanisms that secure the Powpeg. In this way, Flyover provides faster transfers without reducing the security of the system. In case the liquidity provider does not respond, users receive their funds after the regular number of confirmations (in peg-ins) or can cancel the transfer (in peg-outs). Liquidity providers are never in direct control of the user funds, and thus, users are not at risk of losing them.

In the future, we plan to decentralize Flyover even further by making the initial interaction between the user and the liquidity provider on-chain.

References

1. Optimism. https://community.optimism.io/
2. pTokens, https://www.ptokens.io/
3. Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J., Wuille, P.: Enabling Blockchain Innovations with Pegged Sidechains. Tech. rep. (2014)
4. Belchior, R., Vasconcelos, A., Guerreiro, S., Correia, M.: A Survey on Blockchain Interoperability: Past, Present, and Future Trends. ACM Comput. Surv. **54**(8) (2021)
5. Dilley, J., Poelstra, A., Wilkins, J., Piekarska, M., Gorlick, B., Friedenbach, M.: Strong Federations: An Interoperable Blockchain Solution to Centralized Third Party Risks. CoRR **abs/1612.05491** (2016)
6. Garoffolo, A., Kaidalov, D., Oliynykov, R.: Zendoo: a zk-SNARK Verifiable Cross-Chain Transfer Protocol Enabling Decoupled and Decentralized Sidechains. In: Proceedings of the IEEE 40th International Conference on Distributed Computing Systems. pp. 1257–1262 (2020)
7. Gudgeon, L., Moreno-Sanchez, P., Roos, S., Mccorry, P., Gervais, A.: SoK : Off The Chain Transactions. Financial Cryptography and Data Security (2020)
8. Lerner, S.D., Álvarez Cid-Fuentes, J., Len, J., Fernàndez-València, R., Gallardo, P., Vescovo, N., Laprida, R., Mishra, S., Jinich, F., Masini, D.: RSK: A Bitcoin sidechain with stateful smart-contracts. Cryptology ePrint Archive, Paper 2022/684 (2022)
9. MakerDAO: Announcing the Optimism Dai Bridge with Fast Withdrawals. https://forum.makerdao.com/t/announcing-the-optimism-dai-bridge-with-fast-withdrawals/6938
10. McCorry, P., Buckland, C., Yee, B., Song, D.: Sok: Validating bridges as a scaling solution for blockchains. Cryptology ePrint Archive, Report 2021/1589 (2021), https://ia.cr/2021/1589
11. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. Decentralized Business Review (2008)
12. Nick, J., Poelstra, A., Sanders, G.: Liquid: A Bitcoin Sidechain. Tech. rep., Blockstream (2020)
13. Schulte, S., Sigwart, M., Frauenthaler, P., Borkowski, M.: Towards Blockchain Interoperability. In: Business Process Management: Blockchain and Central and Eastern Europe Forum. pp. 3–10 (2019)
14. Singh, A., Click, K., Parizi, R.M., Zhang, Q., Dehghantanha, A., Choo, K.K.R.: Sidechain technologies in blockchain networks: An examination and state-of-the-art review. Journal of Network and Computer Applications **149**, 102471 (2020)
15. Whinfrey, C.: Hop : Send Tokens Across Rollups. Tech. Rep. January (2021)
16. Wood, G.: Polkadot: Vision for a heterogeneous multi-chain framework. Tech. rep., Parity (2016)

17. Zamyatin, A., Harz, D., Lind, J., Panayiotou, P., Gervais, A., Knottenbelt, W.: XCLAIM: Trustless, Interoperable, Cryptocurrency-Backed Assets. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 193–210 (2019)