Background

Terra-Bridge is a protocol for interchain transfers between Ethereum and Bitcoin.

Purpose

The purpose of this proposal is to establish in details the mechanism for permission-less transferring of assets between Ethereum and Bitcoin.

Usecases

Cross-chain transfer between Ethereum and Bitcoin is useful in fields such as:

- Stable coins

: External chain assets such as BTC could diversify the collaterals used in stable currencies on Ethereum.

- Decentralized exchange

: Ability to transfer external assets onto Ethereum expands the available assets for trading.

- Financial derivatives

: Interconnecting blockchain based assets opens up possibilities for various financial derivative products.

Existing works

There are some existing solutions that exists in this field such as the Parity Bridge, POA Bridge. However, these solutions are not meant to bring assets onto Ethereum, but rather scalability solutions for EVM based side-chains. Therefore these solutions are limited to EVM <-> EVM transfers under a permissioned authority based schema.

BTC Relay was one of the first attempts to bring external asset (BTC) onto Ethereum, But it's design limits it's potential to only a one-way relay from BTC to Ethereum, and the costs of relays and lack of proper incentive has rendered the project quiet.

Design Goals

Our goals for the design of the bridge are:

- Two-way transfer support

: The bridge should support two-way transfer of assets between Ethereum and Bitcoin.

- Standardized Interface

: The bridge should follow a standardized interface for easy integration purposes.

- Permission-less

: The bridge should function under a decentralized security model.

Bridge Design

Terminologies

Home

- The chain where we are transferring external assets onto in the form of pegged tokens (i.e Ethereum).

Foreign

- The chain where we are transferring assets from and onto home (i.e Bitcoin).

Bridge

- The system that helps to perform message relay, verification, and hold/release of assets between users on home and foreign.

Validator

- Role that runs bridge client. Similar to miners of traditional PoW chains.

Overview

The bridge is a two way pegging mechanism that works through a rotating validator set and two bridge contracts (“contract” here refers to an executable piece of logic in the context of its residing chain). The bridge contracts resides on the home and foreign chains, hereon referred to as home-bridge and foreign-bridge.

The bridge contracts must be able to accepts and lock funds, verifies cryptographic signature of incoming cross-chain transfer transactions, and release token to user address on successful transfers. The relay of messages between the two bridges on different chains happen in a byzantine fault tolerant way by the bridge validators.

Foreign -> Home Transfer Flow:

[

1460×758 19.4 KB

](https://ethresear.ch/uploads/default/original/2X/5/5f3d996970f165457b785ac9ead9b35dd4582266.png)

1. User deposits BTC into the foreign-bridge. The transaction contains metadata for validators to relay the transfer:
2. T
3. transfer amount
4. R
5. recipient address on home
6. Validators (1 to N) queries and find new incoming transactions on foreign-bridge and send message to home-bridge to relay transfer with the following parameters:
7. R
8. recipient address on home.
9. T
10. transfer amount
11. SIG
12. validator signature of transfer transaction
13. The home bridge receives the incoming message, verifies validator signature and keeps track of the signatures collected from validators. When more than N/2 validator signatures are collected for a given TX

then T

amount BTCT (pegged version of BTC on Ethereum in the form of token) is transferred to recipient.

Home -> Foreign Transfer Flow:

[

1464×710 16.2 KB

](https://ethresear.ch/uploads/default/original/2X/d/d6edee58ff064ff2fe4181de6e9fdc78b0e4420f.png)

1. User deposits BTCT into home-bridge. The transaction contains metadata for validators to relay the transfer:
2. T
3. transfer amount
4. R
5. recipient address on foreign
6. Validators (1 to N) listens for events and find new incoming transaction on home-bridge and send message to home-bridge to collect signatures approving the transfer with the following parameters:

7. R
8. recipient address on foreign.
9. T
10. transfer amount
11. SIG
12. validator’s signature (witness) that will be used to unlock funds on the foreign-bridge
13. Once enough signatures are collected (N/2 + 1) and verified, home-bridge emits event to inform validators the transfer has been fully approved. The last validator to submit the transfer transaction to foreign-bridge along with the collected validator signatures.
14. The foreign bridge receives the incoming transfer request, verifies the collected signatures and T

amount BTC is transferred to recipient.

Number of Transactions for Transfer

Bitcoin -> Ethereum

: 1 initial user transfer request tx on Bitcoin + (N/2 + 1) validator transfer relay tx on Ethereum.

Ethereum -> Bitcoin

: 1 initial user transfer request tx on Ethereum + (N/2 + 1) signature collection transaction on Ethereum + 1 transfer relay transaction on Bitcoin.

Validator Selection Criteria

Through the choice of a BFT consensus mechanism with validators formed from a set of stakeholders determined by depositing stakes in the bridge staking contracts, we are able to get a secure consensus with an infrequently changing and modest number of validators.

Any validators that deposits the stake will become a validator in the next block. Similarly any validator that withdraws from the contract will have their validator status revoked in the following block. A participating validator must register via deposits in the bridge contract on both home and foreign. Since home is Ethereum, the staking and update of validators is straight forwards. For foreign is slightly more complicated due to Bitcoin’s limited functionalities.

Consensus

Any transfer requests with approval from N/2 + 1

validators are finalized. This process continues indefinitely unless N/2

or more validators become unresponsive, in which case the bridge halts.

Security Assumptions

The bridge is secure as long as the attacker controls less than 51% of the validator pool.

Rewarding and Slashing

Validators are incentivized to stay online and contribute to the bridge’s security by relaying transfer requests for transfer fee rewards. For each relayed transfer message, the relaying validator earn a portion of the corresponding transfer fee. At the same time, to prevent validators from going offline without revoking their validator status (i.e withdraw their stake), a slashing mechanism is put in place. If a validator fails to perform the relay for a transfer message, he will get slashed a portion of his deposited stake.

Slashing Condition

The cryptoeconomic security model of the bridge is based on the honest behaviour of the majority (N/2 + 1) of the validators. This also sets the basis for the slashing condition. Given that we collect validator signatures for transfer in both directions of the bridge, we are able to identify on the home-bridge

which validator(s) have missed to participate in the relay of a particular cross-chain transfer. A simple slashing condition could be to punish validators that have failed to submit signatures for any transfers that has already collected more than N/2 + 1 signatures after some unit time of T (could be block number, transfer number, etc).

This means we do not attempt to differentiate between honest or malicious validators from a slashing perspective, but assumes that majority of the validators are honest. Since the default behaviour of validators is to relay all transfers, it will be economically desirable for validators to stay online and not deviate from the default responsibilities.

Home (Ethereum) Bridge Implementation

The home bridge will be written in Solidity and ran on EVM.

The home bridge will have the following interface:

- transferToForeign(string recipient, uint amount)

Events:

- TransferToForeign (string recipient, uint256 value)

Transferring Funds

Dapps or contracts could call the function transferToForeign

to withdraw their BTC from home-bridge to the provided BTC address, and listen TransferToForeign for transfer results.

Home Token

The pegged token used on home chain to represent foreign token aka HomeToken will be a Mintable and Burnable ERC20 token. The mintable and burnable properties or the token allows easy management of the pegged token. It can be transferred to the home bridge and destoryed on transfer outs (to foreign), and created and sent to recipient on transfer ins (from foreign).

Selecting output that could be spent to transfer BTC from foreign-bridge

Transfer funds in bitcoin terms is called: spending of output. In our case, output - is one of the bitcoin transactions that has output pointing to the foreign-bridge. To send money to the recipient, validators must agree on the transaction outputs to be spent and sign them. These signatures are called witnesses and must be pushed to the home-bridge by calling submitSignature(message, witness)

. As home-bridge keeps track of all incoming and outcoming transfer transctions on the foreign-bridge, it can then determine the output that could be used to spend for the given transfer message.

- message

is needed as the identifier of withdrawal request. It consists of the following variables: * recipient

: recipient address

- value

: transfer value

- transactionHash

: transaction where transferToForeign

was called by user.

- recipient

: recipient address

- value

: transfer value

- transactionHash

: transaction where transferToForeign

was called by user.

How to create witness

To get the witness validator must create a transaction with the following params:

- inputs is an array:

[{ 'txid': txOutput['txid'], 'vout': txOutput['vout'] }]

- outputs is an array:

[{ recipientAddress: amountToSend }, { foreignBridgeAddress: change }, { rewardsMultisigAddress: validatorReward }]

One of the possible methods to create the unsigned transaction is execute bitcoind rpc method createrawtransaction(inputs, outputs)

.

Then this transaction must be signed with validator's private key. Possible method of signing the traqnsaction is using bitcoind rpc method signrawtransactionwithkey(unsigned, [validatorPrivateKey], prevTxs)

. The prevTxs

is an array of transactions that outputs will be spent. The signrawtransaction

returns transaction with witness in it. This witness must be submitted in submitSignature(message, witness)

in home-bridge.

Verifying Bitcoin Signature on Ethereum

As part of the transferToForeign

request, the bridge contract needs to be able to verify the Bitcoin transaction signatures (witnesses) collected from bridge validators. Fortunately, we can do this with ecrecover

function in Solidity since both Ethereum and Bitcoin use the same Elliptic Curve Digital Signature Algorithm (ECDSA) for signing. ecrecover

returns ethereum address, but provided signatures are created via bitcoin, thus to verify signature we need to compare ethereum address returned by ecrecover

and bitcoin address of validator. There is a huge difference between bitcoin and ethereum addresses. In order to make a verification, home-bridge needs to store validator public keys and derive ethereum address from it.

Other preparation that needs to be done to pass witness to ecrecover

is splitting it to 3 parts (v,r,s) as described in the Solidity Documentation:

function parseWitness(witness) { signature = bip66.decode(witness); return { r: signature.slice(0, 32), s: signature.slice(32) }; }

Bitcoin witness doesn't contain the v

scalar. v

is needed to identify the part of the elliptic curve the public key must be recovered from, thus we can just sort out 2 possible variants: v = 0x1B or v = 0x1C

.

Foreign (Bitcoin) Bridge Implementation

The implementation of the foreign bridge on Bitcoin will be achieved using a multisignature address. We can achieve this using a M-of-N P2SH multisignature address according to BIP-13. The Bitcoin reference implementation has validations rules limiting the P2SH redeem script to be at most 520 bytes, meaning that the length of all public keys together plus the number of public keys must not this byte limit. For compressed public keys, this means up to N=15.

The redeem script according to BIP-16 can be constructed using the OP_CHECKMULTISIG

opcode in of the format:

[M pubkey-1 pubkey-2 ... pubkey-N OP_CHECKMULTISIG]

In order to increase the upper bound of the validator set we could utilize the SegWit's witnessScript

, which in Bitcoin Core 0.13.1 and up allows up to 3600 bytes of script data. Following BIP-141, we can construct a the P2SH-P2WSH (P2SH with SegWit) multisignature redeem script using the OP_CHECKSIG

opcode in the following format:

OP_CHECKSIG OP_SWAP OP_CHECKSIG OP_ADD OP_SWAP OP_CHECKSIG OP_ADD ... OP_SWAP OP_CHECKSIG OP_ADD OP_NUMEQUALVERIFY

Transferring Funds

Users could use any bitcoin wallet that supports Segwit to transfer BTC to the multisig address.

Validator Rotation

The challenge with Bitcoin is how the deposits can be securely controlled from a rotating validator set. This is done by reaching consensus amongst the existing validators to create and transfer funds to a new multisign completely. When a new validator joins or an existing validator exits. A new multisig is created. The existing validator set than reaches consensus (N/2 + 1) to transfer funds to the new multisig address.

In the future we could also support multiple bridge-pairs for a single currency for additional security.

Credits

Credits to ContractLand Foundation and @KiriKiri for the research and development of this protocol.