Foreign Issued Functionary Asset Emission

John Villar, Christian Moss, Ben Smith
2019-02-22

Abstract

Sidechain technology essentially allows the pegging of an asset on a public blockchain to an analogous asset on a seperate and often more centralized child chain. The benefits of such a peg allows the asset to maintain its supply and market value whilst at the same time allowing the asset to inherit new properties, mainly low transaction fees and fast confirmations. Although sidechain solutions exist, such as Blockstream's Liquid which is based on the original sidechain white paper (Back et al., 2014), they only allow for the pegging of Bitcoin and not other arbitrary assets. In this paper we propose a mechanism for a federated sidechain that allows the emission/pegging of other assets that may exist on a public blockchain such as Counterparty[1] and Ethereum[2] tokens. Furthermore our solution offers the optional ability for functionaries to exercise a higher degree of control over assets, such as the white and blacklisting of assets/addresses, that may be required in certain use cases such as meeting regulatory compliance in certain countries and industries.

1 Introduction

Protocol's allowing the issuance of assets other than BTC on the Bitcoin blockchain have existed for a while. Notably the Counterparty Protocol which was launched in 2014 and was used to issue a range of assets from game items and tickets to digital collectables. As the fees rose on the Bitcoin blockchain it no longer became feasible to use these assets in certain applications which benefit from a low fee environment. Many projects moved to other chains which at the time offered cheaper fees however these chains either also saw fees spike during peak usage or they had questionable decentralization. Furthermore the stability and future roadmap of these chains were not ideal for companies requiring a stable platform and regulatory control. Lastly certain projects benefit from a custom level of decentralization that is not as centralized as a single database but not as decentralized as a public proof of work blockchain.

Users of a certain platform/token will benefit from a low fee environment but may also wish to have the freedom of a fully decentralized public chain, using a peg in and peg out mechanism to and from a public and side chain a user can experience both environments when it is suitable/necessary for them to do so.

[1] Counterparty https://github.com/CounterpartyXCP
[2] Ethereum https://github.com/ethereum/wiki/wiki/White-Paper

2 Design rationale

The technology allowing the peg between a public chain and a side chain is known as a “gateway” a gateway can be described as a group of entities known as functionaires. These functionaries share a multi-signature[3] deposit address and a fixed burn address which are used for the peg in and peg out process respectively.

During the peg in phase a user will send their public chain asset to the deposit address, once confirmed the functionaries will verify the validity of the asset and issue an analogous token on the sidechain. The public chain token is now effectively locked in the multisig deposit address. When a user wishes to peg out the user on the side chain will burn the token by sending it to the burn address, once burned the functionaries will release the locked token that was originally pegged into the deposit address. Next we will explain the peg in process in more detail.

3 Peg in

3.1 Functionary selection

We assume the set of current functionaries are pre chosen by an initiating party and are static, however there can be a process for functionaries to elect new additions to the set via an election process. Assuming that elections of new functionaries happen on the public blockchain it should be feasible for a user to parse the votes from the blockchain and thus derive the current set of functionaries and their public keys.

With these public keys a user can trustlessly generate the deposit address and does not need to worry about receiving a false address. A user can then send an asset they wish to peg in to this address by creating a send transaction with the desired sidechain wallet address added in the op_return output of their transaction.

Once a peg in transaction is detected in the mempool a functionary can deterministically confirm if they or another functionary should be the coordinator for a given transaction. The coordinator can be defined as the functionary who starts and manages the peg in process for a given transaction. This can be achieved by a simple modulus of the integer value of the transaction hash of the deposit transaction. The modulus would return an integer representing a chosen functionary in the given set of current functionaries.

```
/*First alphabetically order the functionaries by their public key and store
in an array*/

functionary 1: pubkey - "fpubkey1"
functionary 2: pubkey - "fpubkey2"
```

[3] Multisignature https://en.bitcoin.it/wiki/Multisignature

```
functionary 3: pubkey - "fpubkey3"

//then using the transaction id which is a valid hex number
e95f1b3277695a44b903d40e9c1ddeeff392b7a35150a0547fec92eb7bf93f68
z
//and the modulus function we can select a functionary to manage the process
var txHash =
parseInt("e95f1b3277695a44b903d40e9c1ddeeff392b7a35150a0547fec92eb7bf93f68",
16)
var functionary = functionariesArray[txHash % functionariesArray.length]
```

Once a functionary is chosen they can then create a transaction that will issue the token on the sidechain and send it to the address specified in the op_return of the peg in transaction. They request other functionaries to sign it, each functionary can sign or return an error if they refuse to sign. If consensus cannot be reached then the managing functionary creates a new transaction to return the token back to the user that the other functionaries can sign instead. Each functionary can have a local list of approved and unapproved tokens if necessary that they can check against before signing the peg in transaction. This effectively allows certain tokens to be blacklisted from the peg in process.

3.2 Shared Verifiable Database

The peg in process requires a database in order to queue the peg in transactions and coordinate the process. For simplicity all functionaries can use the same centralized database as long as this database is verifiable and recoverable.

A single functionary or 3rd party entity can host this database and give read and write access to the other functionaries. This database could consists of a single table with a column for transaction ids and signatures etc
`deposit_tx_id, issuance_tx, signature_1, signature_2, signature_3, issuance_tx_id`

Once a peg in transaction receives the required number of block confirmations to be deemed irreversible in the event of a reorganisation the deterministically selected functionary can insert this transaction id into the database along with an issuance transaction for the peg in and their signature. The other functionaries can check the database and add their own signatures if they approve the peg in. Once all signatures are received the coordinating functionary can then broadcast the issuance transaction on the sidechain and add the resulting tx_id into the record.

3.3 Verifying Transactions

It is important that before a functionary signs a peg in issuance transaction that they can verify that the transaction has not been pegged in previously and the deposit transaction is valid. This can be achieved in the following way

Alice wished to peg in 1 ALICE token, she sends her ALICE token to the deposit address, after it is confirmed functionary A determines that they should coordinate this transaction and adds the transaction id into the database to be queued for peg in.

functionary A uses the tx_id to look up the raw transaction data from the public chain and confirm that the token is a peg in transaction for 1 ALICE token, they also confirm that the transaction has the required number of confirmations to be deemed secure. Once they are satisfied with the transactions status they can create an issuance transaction for the same token and quantity on the side chain. They add this transaction and their signature into the database. Functionary B sees this entry into the database and also proceeds to add their signature into the record. Before doing this Functionary B uses the tx_id to look up the original transaction data in the public chain to confirm it is indeed a peg in transaction for 1 ALICE token and that it has received the correct number of confirmations.

However at this point Functionary B needs to confirm this transaction has not already being issued on the sidechain. That is to say every peg in transaction must be linked to a subsequent issuance transaction on the side chain in a way that any functionary can independently verify.

We can achieve this be requiring that issuance transactions have the txid of the deposit peg in transaction encoded.

i.e.

Alice creates deposit peg in transaction for 1 ALICE token, its txid is "abcdefg", The coordinating functionary A creates an issuance transaction to issue 1 ALICE token on the sidechain and includes the txid "abcdefg" in the op_return output of the transaction.

Now functionary B can use the txid "abcdefg" to

a.) Verify the deposit transaction on the public chain exists and is valid

b.) Check to see if this transaction has already been issued by parsing transactions and blocks on the side chain.

In the scenario that functionary A is a bad actor and tries to create a fake deposit transaction or tries to issue more ALICE tokens than have been pegged in the other functionaries will verify that either

a.) The fake deposit transaction does not exist on the public chain

b.) The deposit transaction has already been issued on the sidechain

Furthermore if the centralized database were to fail or be erased, any functionary via scanning/parsing blocks on the public and sidechain, and finding all deposit and issuance transaction, can recreate and/or re populate the database.

4 Peg Out

Peg out is much the same process as peg in however rather than receiving a deposit transaction and creating an issuance transaction or return transaction the functionaries notice a burn transaction either mine it in a block or wait for it to be mined (in the event that the gateway functionaries are not also the miners in the sidechain) and create a send transaction on the public chain that will send the token that was locked in the deposit address to the user based on the encoded public chain address found in the op_return of the burn transaction. It is worth noting that a user does not need to generate the burn address rather the functionaries can agree on a fixed burn address out of bounds in normal lines of communication as the burn address is not linked to functionary public keys and will likely not change.

5 Side chain construction

The side chain that the assets are issued on is a bitcoin based chain with the proof of work replaced by a multi signing alternative. A set of functionaries which may or may not be the same functionaries in the gateway each sign a block and append it to the blockchain. These functionaries serve a similar role to miners in the bitcoin blockchain chain and are the authority of what is valid in the side chain.

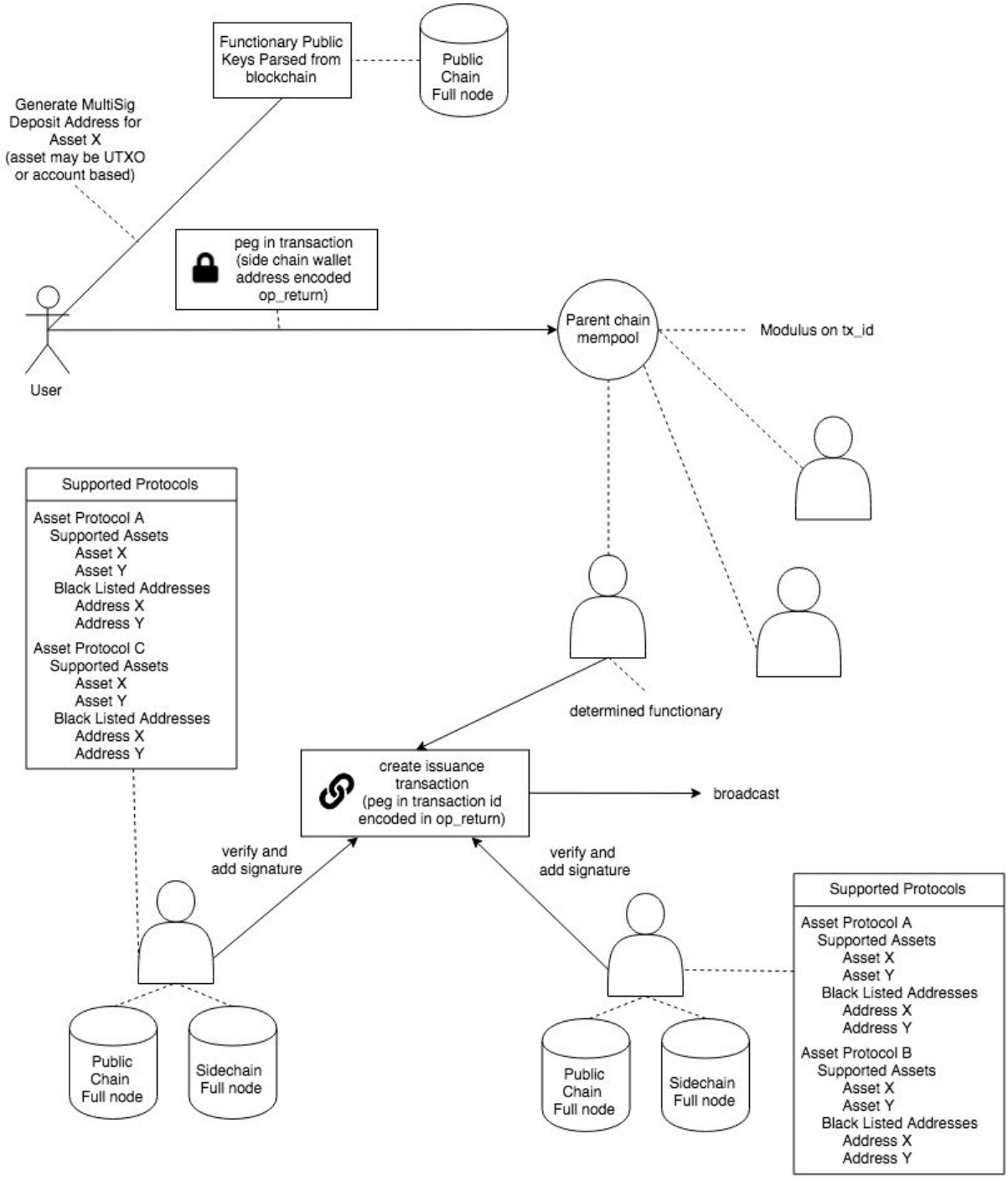
5.1 Native Asset

The native asset of this chain is a utxo based asset similar to bitcoin but it is not pegged to the public chain bitcoin. This asset can be generated infinitely by the functionaries and is used to control spam in the side chain. A user wanting to create a transaction will require this asset in order to broadcast their transaction however the functionaries of the sidechain can supply users with this asset at their discretion.

5.2 Token issuance

All tokens pegged in and issued on the sidechain will follow a similar format to the Counterparty protocol. The Counterparty protocol uses data encoded in an op_return output of a transaction to populate a database showing the current balances of users tokens in the chain. The side chain will instead have an issued asset that is pegged to the public chain bitcoin. Because this asset follows the same protocol as the other issued/pegged assets and is not UTXO based it allows users to also peg in bitcoin to the sidechain as well as allowing for the use of the Counterparty DeX with token to BTC pairs.

6 Schematic



References

(Back et al., 2014) Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille 2014, https://blockstream.com/sidechains.pdf