Background

The state of the art in the design of bitcoin sidechains is to have a federation hold the funds of all sidechain users. A federation is a type of bitcoin multisig address, and the two most popular bitcoin sidechains are Liquid (designed for enhanced privacy) and Rootstock (designed for defi).

To deposit money to either sidechain, users must send money to a multisig address on bitcoin. When you're done using the sidechain, your wallet has to essentially ask the people who hold the keys to that address to please send you however much money you have on the sidechain.

Liquid uses an 11 of 15 multisig, meaning there are 15 keyholders and 11 of them need to agree in order for someone to withdraw money from their sidechain. Rootstock uses an 8 of 15 multisig, meaning just over 50% of the keyholders need to agree.

The problem

The security of a federated sidechain depends on the trustworthiness of the keyholders. In Liquid, if 5 keyholders are compromised, they can at least temporarily halt withdrawals from the sidechain. In Rootstock, the magic number to halt withdrawals is 8. If some of the keyholders are evil and want to send user funds to the wrong destination, Liquid needs 11 keyholders to be evil, while Rootstock still needs 8 (sticking close to 50% makes their numbers kind of simpler).

Using Liquid requires, at minimum, trusting 5 keyholders not to prevent the other keyholders from honestly performing the withdrawal function, and using Rootstock requires trusting that there are 8 honest keyholders. One of my goals in the sidechain design space is to think of ways to reduce this trust requirement. Ideally, I'd like to reduce it to 0, so you do not have to trust any of the people who process withdrawals from a sidechain, but I don't currently know a good way to reduce the trust requirement all the way down to 0. Instead, in this document I'll outline a way to reduce the trust requirement to 1: you only need to trust 1 person in the federation to be honest, thus your withdrawals are likely to go through unless *every* federation member is compromised. I think that's a meaningful improvement over the current state of the art.

Shimchain

Shim stands for Single Honest InterMediary. The idea starts with a federation *all of whose* members need to agree before any transaction can happen. (If that sounds like a step in the wrong direction, since I want to get to 1 not all, keep reading!) Like with existing models, a shimchain federation has 15 keyholders, but it uses a 15 of 15 multisig, which means no money can move without unanimity among the keyholders. To deposit money to a shimchain, you send money to that 15 of 15 multisig, and to withdraw money from a shimchain, you ask the 15 keyholders to send it back. But here's the difference: *before* you make your deposit, the keyholders give you a signed transaction that sends you your money back after 2 weeks. They can do this using bitcoin’s timelock technology: a transaction can be created that isn’t valid *right now* but *will* be valid in two weeks.

With this timelocked transaction in hand, you can send money into the 15 of 15 address and know that one of two things will happen: (1) you’ll get your money back in two weeks or (2) all 15 keyholders will agree to send it somewhere else. If nothing else happened, that would accomplish the goal: the keyholders cannot steal your money unless *all* of them agree to steal it. As long as 1 keyholder is honest, no theft can occur.

Now, a shimchain would be pretty useless if all you could do with it is deposit money, wait two weeks while nothing happens, and then withdraw your money. A useful sidechain lets you *transfer* your money to other people in ways that are difficult on bitcoin. For example, you can do a confidential transfer on Liquid, where no one can see *how much money* you sent, and you can do a conditional transfer on Rootstock, where your transfer only happens if a smart contract says it should (and there are various kinds, such as lending contracts, gambling contracts, and many, many ponzi schemes).

I think a shimchain can allow transfers via the following mechanism. Suppose you deposited 10,000 sats to a shimchain 1 week ago and now you want to transfer 5,000 of them to Bob. First, create your shimchain “transfer” transaction, put it in the shimchain’s mempool, and wait til a shimchain “proposer” (equivalent to a bitcoin miner) adds it to a shimchain block. But this block isn’t mined yet, it is only proposed. Anyone can propose a block, but they are only mined when this happens:

The shimchain keyholders look at all the transactions in the block, check if they all follow the consensus rules of the shimchain, and, if they do, they all agree to create a bitcoin transaction that sends all the money in the 15 of 15 address out of it and back in again. This transaction *invalidates* the transaction they gave you when you made your initial deposit. You can’t withdraw your money using that transaction anymore. But before they do that, they give you a *new* transaction, signed by all of them, based on the new situation: you and Bob will *both* get 5,000 sats after one more week.

Just like before, all of this only happens as long as all of the shimchain keyholders say it should. If all of them are honest, the new block is mined, and everyone (including you and Bob) can withdraw their money when the time comes. If all of the keyholders are dishonest, it’s bad: they can send your money wherever their evil hearts desire. If only some of them are dishonest, it’s not super bad. You tried to send money to Bob, but one or more of the keyholders was dishonest, so the block never got mined. Your transaction effectively failed, but you can still withdraw your money after one more week, and just pay Bob with regular bitcoin instead.

Details

I suspect proposers and keyholders will be the same people. The keyholders themselves will propose, and then mine, blocks containing sets of transactions signed by users of the sidechain.

For this service, they will want to take a fee. Like with bitcoin, users can set a fee in their transactions by ensuring the amount in the “outputs” is smaller than the amount in the “inputs.” Thus keyholders may keep a little slice of every transaction. If your fee is too small for their taste they can refuse to mine your transaction and maybe tell you what fee rate they’ll do it for. If you don’t like their fee rates, just withdraw your coins when your time comes – they can’t stop you unless all of them agree to steal from you.

Some types of transactions may require more trust in keyholders. For example, suppose you want to send Bob 1 btc via a smart contract that only lets him have it if he sends you 5 monkey inscriptions. Basically a uniswap clone. Suppose you send the 1 btc to the smart contract in block 5, and it gets mined, and then Bob sends you the inscriptions in block 6, and it gets mined, and then Bob withdraws the 1 btc from the smart contract in block 7, but *right then* a dishonest keyholder refuses to mine block 7. 2 weeks go by and you withdraw your inscriptions, but Bob cannot get the 1 btc because the miners never mined block 7 and thus never gave him a signed transaction sending him his money. No fair! Uniswap clones on a shimchain must prevent this through some sort of atomic swap technology, otherwise Bob has to trust that a keyholder won’t “go rogue” right in the middle of a series of multi-block transactions.