I'm uploading this as an edit/repost of an older write-up available here:https://hackmd.io/s/Hk-gS-UOG

This document attempts to extend Vitalik'sMinimum Viable Plasma so that a third party ("exiter") can be incentivised to execute exits on behalf of a user on the network. Generally, this is accomplished allowing a third party to exit on a user's behalf in exchange for a fee. The third party in this design is trusted

.

Delegated Exits

Delegated Plasma exits are a useful construction. It's likely that it'll generally be hard to get every

user on a Plasma chain to submit exits if blocks are being withheld. This might occur for any number of reasons. For example, a user may be unable to exit if they are being targeted and deliberately denied service/connectivity to the root chain during the exit period.

Fees

However, simply allowing others to exit on a user's behalf isn't enough. It's not reasonable to expect that others will behave altruistically, so it's necessary for the user to specify a fee for this service. As it's probably poor user experience to have users determine a fee for each of their UTXOs, it's simpler specify a flat % fee for every exited UTXO. This could be maintained as a mapping (address => uint256)

where each address

maps to a uint256

fee to be divided by some uint256 FEE_BASE

to calculate a fee %.

Unnamed Exiters

A few possible constructions exist when deciding who can actually perform exits on behalf of a user. An "ideal" design allows any user to exit on behalf of any other user. However, we want to make sure that exits are only performed when actually necessary. Otherwise, an exiter might be incentivised to trigger an exit as quickly as possible in order to capture the exiting fee.

If we can prove that a chain is actually suffering from an "exit condition" (e.g. block withholding), then this problem is easy to solve. It'd be quite easy to require some initial deposit and slash the deposit if an exit is made when the chain is not under exit conditions. Even so, we might want to allow exiters to exit for us under normal chain conditions.

Unfortunately, it's not trivial to prove that the chain is actually under exit conditions. A naive solution may require child chain blocks be submitted once every X parent chain blocks, else the chain is under exit conditions. Extra-protocol interactions might cause the network to experience these conditions temporarily even if users on the network understand this is temporary. In general, the subjectivity of exits on the network make them difficult to quantify.

This subjectivity might be addressed by implementing a parent chain vote to determine if the child chain is under exit conditions. Users on the network would signal one way or the other, dependent on their % assets owned in the parent contract. This may end up being ineffective for several reasons. If the underlying asset on the parent chain derives its value from the Plasma chain, then large stakeholders have an incentive to vote that the chain is not

under exit conditions to prevent a mass exit (and a rapid decrease in value of the asset). If the underlying asset is something like Ether, then majority stakeholders can grief the network by signalling exit conditions at no significant cost.

We could prevent some exit griefing by implementing another transaction that allows a user to "cancel" an exit called on their behalf, but this has its own issues. If we don't require the exiter to place a deposit on the exit, then it's in the depositor's best interest to repeatedly attempt to exit until the total transaction cost on the root chain exceeds the potential exit fee. If we do require a deposit, then an active user might cancel the exit and call the exit themselves in order to save the exit fee at no additional cost (although this behavior might be necessary and preferred).

Named Exiters

A (likely) better delegated exit model is to require users to specify a named exiter or list of exiters. The root contract could maintain some mapping (address => address)

or (address => (address => boolean))

that specifies a user's permitted exiters.

This is effectively a trusted subset of the “Unnamed Exiters” design. We inherit some problems, but we gain some useful properties. Most importantly, we effectively build a reputation system where certain named exiters are identified as “trustworthy.” Users will likely name exiters who have a history of acting in their customers’ best interests. As a result, exiters have an incentive to behave.

This design isn’t perfect. It’s still possible for exiters to attempt to exit against the user’s will. However, it’s easier for a user to mitigate this attack by removing the exiter and then cancelling the exit. The worst case scenario allows an exiter to “exit scam” (confusing terminology, oops) and call exits for as many of their customers as possible in the hope that a few are inactive.

Named exiters should only expect to realize fees from inactive users. Active users have an incentive to cancel the exit transaction and exit for themselves. Exiters will only participate in this system if the total fees generated from inactive users exceed the transaction fees of creating the exits on the root chain. This becomes more feasible if mass exits are implemented. Exiters will probably wait some short period of time before submitting an exit to allow active users to submit exits first.

Rational exiters should only call exits for users with a sufficiently high transaction fee. A fee market will probably emerge over time as users and exiters decide on some equilibrium fee.

Unnamed Exiters + Pre-signed Transactions

This idea is Joseph Poon’s from a conversation a few days ago:

Another way to achieve a similar result to the idea of named exiters is to have users pre-sign exit transactions for each of their UTXOs and to pass those transactions to some trusted parties. This would push more of the requirements onto the client-side (less interaction with the root chain), but would require that users create/send new exit transactions for each new UTXO. This might be problematic if the user receives some UTXOs after they go offline and therefore can’t sign/send the exit transactions.

Exiter Pools

It may be possible that “exiter pool” smart contracts come into existence. These contracts could mitigate some of the risks of named exiters by requiring members vote on which exits to submit. This is probably extremely inefficient and would likely require centralized management. Fees from exits would then be distributed to members relative to their investment in the pool.