The mechanism of encumberments has come up under different names in cross-shard communication and plasma. We argue that recognizing this allows us to systematically construct better mechanisms, e.g. channels-on-plasma with short dispute times, and argue for plasma dispute times to be much longer than they are currently planned to be.

Encumberments

The techniques I am refer to can be found inEncumberments: instant cross-shard payments over slow cross-shard base-layer communication as a layer 2 as well as Simple Fast Withdrawals. Here is my attempt at a self-contained explanation highlighting their commonalities. Both start with base-layer-operations that are slow for some reason (cross-shard transactions and plasma withdrawals respectively). This normally result in bad UX (user has to wait a long amount of time with their money stuck in the slow mechanism), which they try to mitigate. The mitigation mechanism uses a common property: an external observer can objectively determine the outcome of the base-layer-operation faster than the base layer itself can (e.g. an observer can objectively determine the outcome of a plasma exit quickly if the chain is available). Note that this property must be carefully engineered in: for instance, the plasma contract must ensure that blocks roots submitted after an exit is initiated cannot be used in the exit game, otherwise this property no longer holds.

In the case that this property does hold, an “encumberance” is used where the beneficiary of the base-layer-operation is set to an entity that is restricted in a specific way. In simple fast withdrawals as written this is basically a smart contract whose ownership right is treated as an NFT; in the cross-shard framework this is an account with an in-protocol “encumberment queue”. The restriction in necessary in that both mechanisms fail if the beneficiary is an ordinary ETH 1.0 EoA.

Encumberance Chains

Simple encumberments as described can change their own beneficiary, but they can also be composed by setting the beneficiary to another encumbered entity. In a chain of composed encumberments, the outcome of the base-layer operation is to send ETH to some encumbered entity, which sends it to some other encumbered entity, and so on, where anyone can quickly convince themself that the last entity in the chain is the ultimate beneficiary even before base-layer operation resolves. This does require protocol-level support (e.g. the encumberence queue) to be really useful; if the ultimate beneficiary is an ETH 1.0 EoA, that account cannot convincingly transfer ownership to someone else and the chain cannot be extended. An alternative to protocol-level support is for most users to stop using EoAs directly, instead using single-owner multisig contracts that can irreversibly encumber themselves (irreversibly set their beneficiary to some other entity).

Channels with Short Dispute Lengths

Note that channel disputes do not enjoy the property that “external observers can determine the outcome of [the dispute] before it is over”. However, a finalized (closed) channel has this property. This is useful if the channel is funded by a plasma deposit which has a dispute time longer than the channel’s (e.g., for instance, a plasma dispute period of 2 months and a channel dispute period of 1 day) and hence finalizes/closes without being funded.

The end result is that the beneficiary of the plasma exit is set to the state channel smart contract on ethereum. That smart contract engages in a 1-day-long dispute over who its beneficiary is; at the end of that 1 day, it irreversibly encumbers itself to send all received money to that beneficiary [1]. 2 months later, the withdrawal succeeds, the ETH goes to the finalized channel contract, and then continues on from there.

Payment Channel Networks, and an argument for long plasma dispute time scales

The above logic continues to hold if the channel-on-plasma is an intermediary channel in a payment channel network. Hence a payment channel network can transparently use either channels-on-ethereum or channels-on-plasma for the intermediate links.

A unified encumberment mechanism is also useful if participants tend to hold their funds in multiple mechanisms at once. For instance, a user withdrawing from a typical well-behaved plasma chain will probably do so via a cross-chain atomic swap or a channelized cross-chain atomic swap.

In the long term, if a protocol-level encumbermen mechanism becomes commonly used, setting for the plasma dispute time scale to be very long (e.g. 2 months instead of 7-21 days as currently envisioned) becomes possible. In an ideal model where “objective verification”, i.e. downloading a plasma chain’s blocks and computing the outcome of a dispute, is costless and always possible, the plasma time scale does not matter. However, this is not a realistic assumption; for e.g., verifying the state of 100-hop payment channel for which each intermediary channel is a channel on plasma which has finalized on ethereum but for which the plasma withdrawal is still in progress requires downloading block data of 100 plasma chains, which can be prohibitively expensive. Similarly, users have to download shard data for all shards which participate in an encumbermance chain.

The upper limit of the plasma dispute time scale [2] will then be set by the costs of doing such verification, as well as by usage patterns (how often payments are sent, etc).

Thanks to @gakonst @liam @technocrypto for discussion and coming up with this, and@kfichter @vbuterin for writing up the linked posts. The transitive thankees of those posts are hereby thanked as well.

Notes:

[1] There is a slightly more complicated alternative where the finalized channel contract changes the beneficiary of the plasma withdrawal.

[2] Also the shard cross-link frequency, in the case that we only support payments.