The GPACT protocol (paper, repo) allow atomic crosschain function calls across blockchains, sidechains, privacy groups, and rollups. The technique can be viewed as a two phase commit protocol, where the call execution tree of (blockchain id, contract address, ABI encoded function call data)

are constructed into a data structure and committed to.

A concern operators of private permissioned blockchains has been is that the call execution tree is advertised to participants of a private blockchain that should not have access to it. For example, imagine a trade-finance application in which there is a trade blockchain which holds logistics information (who owns what goods and where are they), and there is a finance blockchain which holds financial information (who owns what tokens). When goods are shipped, a function call might occur on the trade blockchain “goodsArrived(uint256 _numberOfGoods)

”. This function would update who owns what, and might then read the agreed price given the business terms, and then call “transfer(address _from, address _to, uint256 _amount)

” to transfer money from the buyer to the seller. On the trade blockchain, it is OK that the call execution tree is available to everyone, as they can see the transaction and determine the outgoing function call. However, on the finance blockchain, there is no need for the participants of the blockchain to know how many goods have arrived. Additionally, the information about how much each item is worth is leaked to participants of the chain.

An additional concern is publishing the call execution tree before it is executed could facilitate front running attacks.

The proposal to mitigate both of these issues is for the user to commit to a call execution tree that only contains Salted Hashes, instead of (blockchain id, contract address, ABI encoded function call data)

. When the user submits a Segment or Root transaction, they can then submit the SALT and the (blockchain id, contract address, ABI encoded function call data)

for the function to be called, and any outgoing function calls that are made by that function.

@hmijail suggested a further improvement of creating a Merkle Tree like structure for the call execution tree and committing just to the Merkle Root of the data structure.

This will be implemented in the coming months in the GPACT repo as an alternative to GPACT and SFC. My gut feel is that the changes from GPACT will be gas cost neutral.