It would be useful for a light client, especially in a high value application like a bridge, to verify finality using Casper FFG, rather than just the sync committee. In particular, accountability gives some security even against 1/3 or 2/3 of the validator set being malicious. Honesty of validators is a less reasonable assumption when considering external protocols unrelated to the proof of stake token, such as bridges.

Accountability is defined as if two full clients of the protocol see blocks on different forks as finalised then the union of their views can be used to prove that 1/3 of validators misbehaved.

In Accountable Light Client Systems for PoS Blockchains, my co-authors and I consider an assymetric extension of this accountability property to light clients. Essentially, we consider an accountable light client protocol as one where if a full client and a light client see blocks on different forks as finalised then the union of their views can be used to prove that 1/3 of validators misbehaved. The equivalent property need not hold between two light clients. In that paper we used it for light clients that do not know the validator's public keys, but we can use the same ideas to skip other checks full nodes perform.

There a couple of annoying things about verifying Casper FFG:

1. We need to verify a chain of supermajority links between justified checkpoints
2. We need to verify for each link that the block at the end of the link is actually a descendent of the block at the start

Why these are annoying for a light client:

1. Verifying a supermajority link isn't cheap. It can require checking 4096 aggregate signatures from 500,000 public keys. This happens every 6.5 minutes. On the other hand the validator set doesn't change fast so maybe if we only want to have an update every day, it would be good for it to be ok to only prove that one or two links were signed.
2. This means verifying every header. Again if we only want to update finality occasionally, this is some overhead.

The problem is that if everyone skipped even one of the checks 1. or 2., we'd be able to finalise two checkpoints, b and b', on different chains.

If we skipped 1., there needn't be a chain of links and so we could have a situation like:

b ----------___/

___
/ \

----------b'

(This diagram assumed that 1 link, one of length 1 from b

is enough to finalised b

, but a light client checking any constant number of links would have the same problem.)

If we skipped 2., links could jump from one fork to the next like:

b ----------____/ ___/ \ \ \ \ ___ \ / \ ----------b'

In neither case do these links violate the slashing conditions.

However proofs that skip 1. or 2. can be secure in our assymetric model i.e. if everyone doesn't skip them. Full nodes following Casper FFG do not skip 1. or 2… So we can define "full proofs" and "light proofs".

A full proof of the finality of b'

consists of a sequence of justified checkoints r \rightarrow b'*1 \rightarrow b'_2 \dots \rightarrow b'_m=b' \rightarrow b*{m+1}

with h(b_{m+1})=h(b')+1

and headers that show for every link, the starting checkpoint is an ancestor of the ending checkpoint. Full nodes have full proofs of checkpoints they see as finalised (or later checkpoints on the same chain).

A light proof of the finality of b

is two supermajority links, one b_{start} \rightarrow b

and one b \rightarrow b_{end}

with h(b_{end}) =h(b)+1

. There is no context about ancestry or evidence that these checkpoints contain blockhashes of blocks at the claimed height.

I claim that:

Suppose that there is a full proof of a checkpoint b'

and a light proof of a checkpoint b

such that b'

is at least as high as b

but the root(blockhash) of b

does not correspond with the root of the block header at the height of b

in the chain of block headers containing b'

in the full proof. Then the union of the full proof and the light proof contains pairs of votes violating the slashing conditions from >1/3 of validators.

With this, light proofs of blocks that do not get finalised will be able to cause validators to get slashed, provided that

- finality does not stall from the point of view of a full node
- the light proof is public e.g. if it on another public blockchain.
- the light client does not accept proofs from far into the future, which the chain takes so long to reach that many validators could have exited and withdrawn.

Then a full node seeing that a different fork was finalised, that reaches at least the same height, will be able to use their own full proof and the light proof to generate proofs of misbehaviour.

Proof of the claim:

We need to find a supermajority link from full proof and a supermajority link from the light proof such that any validator signing both would violate a slashing condition. Since >2/3 of validators signed each, more than 1/3 signed both and can be slashed.

Consider, the link from the full proof whose starting height is under h(b)

and whose end is of height at least h(b)

. There is such a link because h(b') \geq h(b)

and so there is a least i

with h(b'_i) \geq h(b)

. Then this link is from b'_{i-1}

to b'_i

There are several cases, depending on the height of the checkpoint at the end of the link, h(b'_i)

:

- h(b'_i)=h(b)

. The links b_{start} \rightarrow b

in the light proof and b'_{i-1} \rightarrow b'_i

from the full proof have different endpoints of the same height since unlike b

, b'_i

is an ancestor of b'

and so b \neq b'_i

. Then validators who signed both are violating Commandment I by equivocating at height h(b)

- h(b'_i)=h(b)+1

. The light client proof also contains the link b \rightarrow b_{end}

to a checkpoint of this height. These two links have different starting heights so they cannot be the same link. Then validators who signed both are violating Commandment I by equivocating at height h_{mid}+1

.

- h(b'*i)>h(b)+1=h(b*{end})

. The links b \rightarrow b_{end}

and b'_{i-1} \rightarrow b'_i

have h(b'*{i-1} < h(b) < h(b*{end}) < h(b'_i)

. Validators who signed both links are violating Commandment II by voting within a span.

One of these possibilities must occur, so over 1/3 of validators have votes violating a slashing condition as required.

It would be nicer to have a light proof containing only one link, the link of length 1 starting at b

. The link ending at b

was required to deal with the h(b'_i)=h(b)

case. We could avoid needing this, and so getting a one link light proof, by adding an extra commandment making it illegal to sign two links starting with different checkpoints at the same height.