Answers to reviewer's 606A questions:

1. *CKS properties interpretation: Perfect completeness:* If all of CKS.Verify inputs except for an alleged aggregated signature have been generated honestly and if the alleged signature is accepted by AS.Verify, then CKS.Verify on the alleged signature and honest inputs accepts; it is the counter-notion to AS perfect completeness for aggregation (Definition 3.1); it is used for proving (Section I) perfect completeness (Definition H.5) of our light client instantiation (LCI), Section H.3.
   Soundness: An adversary cannot output a CKS verifying proof and alleged aggregated signature pair without the alleged aggregated signature being accepted by AS.Verify; it is crucial for proving (Section J) LCI accountability completeness (Definition H.7).
   Unforgeability: Is inherited from the underlying AS and is not directly used for proving any specific property, but is a consequence of AS unforgeability and CKS soundness; it shows CKS has more security properties beyond an argument system. AS unforgeability is used for proving LCI accountable soundness (Definition H.7).
2. Polynomials in Section 4 are Lagrange interpolations over a cyclic group of values chosen as: b(X) - the bits in bitmask; pkx(X), pky(X) - all public keys' x and y coordinates, respectively; kaccx(X), kaccy(X) - the x and y coordinates, respectively, of the iterative sum of the actual signing validator's public keys. Intuition for polynomial identities: id1, id2 ensure the x and, respectively, y coordinates of the partial sum of actual signing validators public keys follows formulas (*),(**) from Claim 4.1; id3, id4 ensure first partial aggregate sum is h and the last is h + apk; this is necessary in order to ensure the addition of the public keys (which are elliptic curve points) never falls into case c) defined in Claim 4.1; id5 ensures b(X) evaluates to bits over the chosen cyclic group.

Answers to reviewer's 606B comments:

1. The accountability of our light client construction is relying, in turn, on the accountability of the underlying consensus protocol we build upon. The underlying consensus accountability is crucial for proving (Section J) our accountability completeness property (Definition H.7). The accountability of the underlying consensus intuitively works since/if the dishonest validators can be found and slashed for their faulty actions.
2. We can easily generalize our light client to validators with unequal stakes: public inputs to our SNARKS should additionally include commitment to stakes (integers) and sum of all stakes; aggregated public key is replaced by scalar product between stakes of each epoch's signing validators and their public keys; public bitmask ensures accountability of our light client instantiation is preserved.

Answers to reviewer's 606C weaknesses/reasons to reject:

1. Throughout the submitted version and additionally in the appendix we go to great length to formalize a general approach to our problem, starting from aggregatable signature scheme, committee key scheme but also including an extension to the standard PLONK compiler and defining a very detailed and general accountable light client system together with the corresponding security properties.
2. 2^20 represents not half but approximately twice the current number of Ethereum's validators. Our custom SNARK(s) prover time for input size 2^19 is around 4 minutes on an average commodity laptop without any particular SNARK-friendly hardware optimisations. In appendix I we additionally include a discussion on how to extend/apply our work to Ethereum.