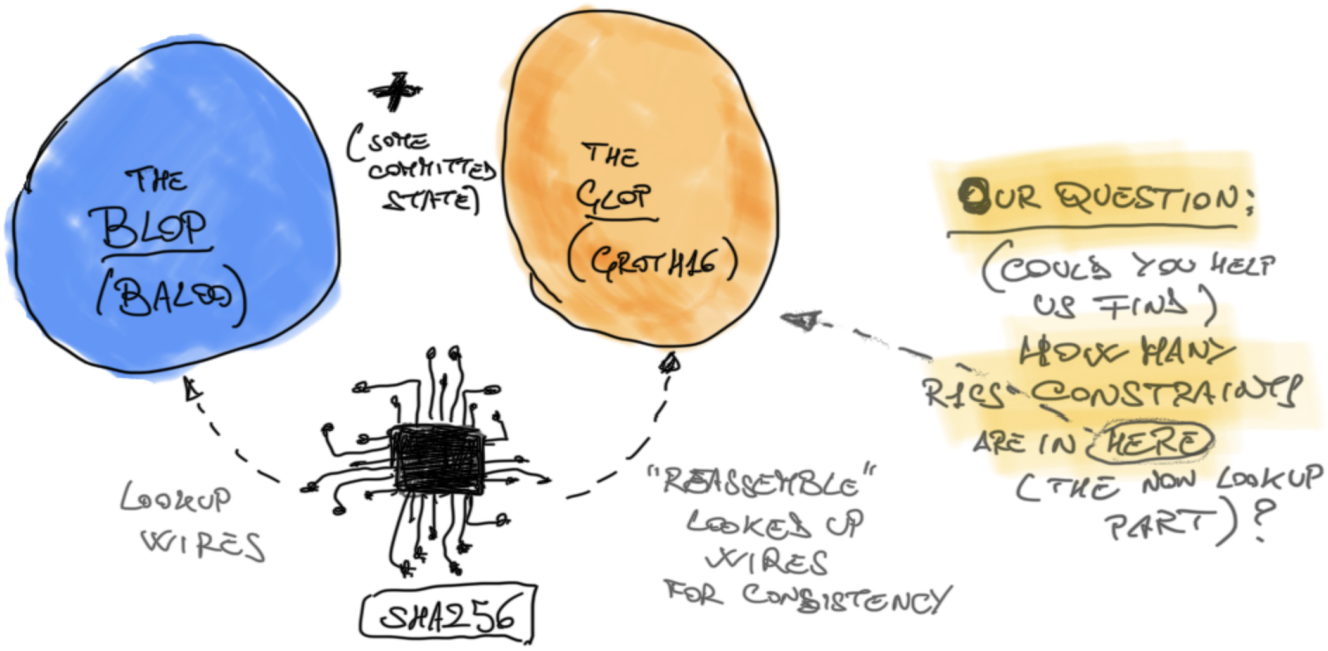
We have been looking at how efficient it would be to apply **Baloo-style lookups** to SHA256.

The problem blocking us:

We have some idea of how many lookups there would be and their cost in Baloo. Lookups in SHA have been documented for example here and here. What we are unclear about is what those descriptions imply for what is left to prove “besides lookups”.



More on the setting

We are interested in particular in using a Groth16+Baloo approach as follows:

- *Separate all lookups in SHA from the constraints that show these lookups are “assembled correctly to give us the SHA function”*. (I’ll give these two parts two very serious sounding names so we can clearly reference them later). Then have:
- a ***blop*** step: apply Baloo to prove the lookups in bulk
- a ***glop*** step: prove all the other constraints using Groth16 (it could be a different proof system. At the moment we are looking at Groth16 first)

(Not crucial here, but maybe worth clarifying: for this to work, naturally, the blop and the glop proofs need to share some state. This state will consist of all the “wires” we looked up through Baloo, committed through KZG)

Our problem, alternatively phrased:

What is the rough number of constraints proven in the *glop* step? We are unable to extract them easily. Could you help us with it? See also below.

More details, the blockers, clarifications :

- We know how many lookups to do approximately from this description of SHA256 as implemented in Halo2 [SHA256 chip in the halo2 book by ZCash] or from [Dmitry's description]
- We do not know how many more _R1CS_ constraints are left. Notice that the constraints as they are described in the doc are in a different arithmetization: not Groth16's but a Plonk-ish one. Someone understanding those descriptions in detail might be able to extract some type of conversion. We, sadly, could not do that easily. :)

Maybe somebody can help us extract those numbers from those docs?
If you have other hints, that is also appreciated. Thanks!

Misc comments:

- Above I mention Baloo, but it could be any "standalone" lookup scheme, even, e.g. the more recent "cq".
- Since we are sharing this state between blop and glop, glop will need to be a LegoSNARK-ish (commit and prove) version of Groth16.
- To clarify the problem of different arithmetizations a little bit: if one takes, for example, the description in Dmitry's doc, everything that is not a lookup seems to be taken care by the permutation check. In glop (Groth16) we will have to do that through equality checks among some of the wires.