Taiga currently does not have function privacy, but it will. The primary goal is to ensure the appropriate predicate(s) are satisfied in a transaction without publicly revealing which predicates they are. There is also a secondary goal of improving efficiency and scalability by potentially accumulating predicate proof verification. There is also a secondary goal of preparing for Taiga rollups.

The initial plan for function privacy is to build an accumulator circuit that takes as input:

- Blake2s commitments to the required predicates
- Commitments to the public inputs of those predicates
- One or more accumulators

and ensures that predicate proofs are accumulated correctly. Currently, the plan is that every predicate circuit will share the same Halo2 configuration (an assumption that can be revisited later).

Function privacy depends on the following Halo2 issues:

- Poseidon duplex gadget - partly working, requires comprehensive tests
- Transcript gadget - implementation begun, but not working yet. Will revisit now that better documentation is available
- ECC gadget - Support full-width scalars for variable-base mul probably will be straightforward, but requires some work
- Implement the recursive proof verifier It’s likely that Taiga can simplify some steps over the full recursive verifier/IVC to get the lesser goal of function privacy

Function privacy also depends on the Blake2s hash and commitment gadgets that are currently being implemented.