In order to support contract upgrades, we need a way to store what the current implementation is for a given contract

such that it can be accessed from a private execution and doesn't introduce contention between multiple txs. This requirement is independent of how

we implement contract upgrades: whether we enshrine it as part ofcontract classes, or implement it at the app layer using proxies and delegatecalls, since we need to read the current implementation address

from somewhere.

Note that this problem only affects private functions, since public functions can just read that information from public state, knowing it's up to date.

We'll summarize here some of the approaches discussed inthis thread.

Slow updates tree

This seems the most promising solution, since theslow updates tree is designed for exactly this use case: data that rarely changes but needs to be accessed in its most up-to-date version from multiple private executions. Note that if we enshrine contract upgrades, then we'd need to enshrine the slow updates tree as well.

A drawback of using the slow updates tree is that upgrades cannot be executed immediate, but can only be scheduled to be executed in the next epoch change - which is not a bad thing per se. Another drawback is that now every single tx that goes through an upgradeable contract has its validity restricted to a slow update epoch, which could be annoying for UX if a user attempts to send a tx close to an epoch end.

Unencrypted private notes

Storing the implementation address in an unencrypted (ie publicly readable) private note (ie a piece of data in the private data tree) doesn't work, since we have no way to prove that the tx is referring to the latest version of that data without nullifying it. And if we nullify, then every other tx that needed to go through that same contract becomes invalid.

Call to a public function

Another non-solution is adding a call to a public function with the implementation address used during private execution, and having the public function verify that that address matches the latest one stored in public state. However, this leaks privacy, since it becomes public what contract was executed. This problem goes away if we decide to forgo this privacy guarantee, as some private chains do.

Use FHE magic

Based on recent discussions, it seems it'd be viable to have shared private state using FHE. But FHE is already fairly expensive on its own, and creating a ZKP over FHE becomes prohibitive, in particular if we expect end-users to be generating these proofs.

No upgradeability

Just solve the problem by not supporting upgrades, or at least not supporting upgrades for private functions

. Whether upgradeability is a feature or a bug is still a point of discussion in Ethereum, I personally think there's value in allowing communities to upgrade their contracts for iteration or bugfixing, based on their governance process of choice, so I wouldn't go down this route.