The PR to introduce Eth1 withdrawal credentials had been merged and it's clear it'll be implemented in not so distant future.

There's a number of protocols and stakers who would benefit greatly from the possibility to change their BLS withdrawal credentials to Eth1 EOAs/smart contract. Collectively they account for at least 10% of staked Ether, maybe more, and that share is growing. Lido alone has >5% of all Ether staked and accounts for >20% share of all deposits these days The withdrawal credential rotation feature can reduce the amount of counterparty risk in eth2 staking and make the existing liquid staking protocol more secure.

I see three possible ways that could be implemented, in no particular order of preference:

Withdrawal credential rotation state transition function embedded in eth2 protocol

We can design a transaction-like method by which withdrawal credentials can be rotated at any time if authenticated by the current withdrawal credential holder. I don't think it's a good idea, because it's a lot of additional complexity for the protocol and the client, and that feature had been discarded in the past for good reasons, both technical and economical in nature (e.g. it enables a eth2 validator market).

One-time withdrawal credential change from BLS to Eth1

A variant of the previous option, but much more limited in scope. Only one change per validator is permitted, and it's only from BLS credentials to Eth1 credentials. That one is easier to implement and has fewer economical implications. One possible mechanism would be to allow block producer to put a signal to change its withdrawal credentials in the block it makes, signed by the existing withdrawal credentials: that neatly avoids the hassle of implementing transactions on the base layer for now.

One-time bulk change of withdrawal credentials during a hardfork

During the preparations of the next hardfork, or one after that, we can collect all the signals to change BLS withdrawal credentials to Eth1 ones without processing them (e.g. using graffiti field or eth1 calldata for ordering), and incorporate one bulk state transition in the hardfork itself. That would be a bit harder than the previous option coordination-wise but will allow making state transition function a one-time thing instead of a part of a protocol.