Motivation:

In current ERC20 smart contracts a loss of the private key means an irrecoverable loss of the ERC20 tokens associated with an address. This fact scares non-technical users. This text describes a simple user-friendly mechanism to recover access to tokens.

Idea:

In its simplest form the owner chooses a password and stores a double-hashed version of the password in the smart contract. Anybody can then send a commit with the address he request funds from, a certain amount of tokens as a stake and the hash (not the double-hash) of the password. There can only be one pending commit per time frame. Then the person who successfully commits must reveal the clear text password. If the clear text password matches the commited hash and the double-hashed version stored in the smart contract he can access funds. If he doesn't reveal or the provided clear text password doesn't match, the provided stake is lost forever.

Analysis:

A commit reveal scheme is obviously necessary to prevent a miner from extracting the clear text password from a transaction, discard the transaction and send his own transaction to request access to funds. A miner has however no incentive to imitate a commit transaction because he is unable to reveal – he doesn't have the password and would loose the required stake. When the user successfully commited nobody else can commit because there is only one commit per time frame. Then the user has the possibility to reveal in a certain time frame and nobody can stop him from doing so.

Obviously an attacker could double hash many different common passwords and match them with double hashes stored on the contract. To prevent this the user must have a long enough password. However one can trick here. The user can just have a "common" password and append a personal unique identifier like FirstNameSecondNameBirthplaceParentNames .

This unique identifier acts a source of randomness. There is not additional burden because people remember these things and they can even be public. It makes it economically infeasible for an attacker to simply hash through all "common" password because he would have to do it for every single person.

One can also simply steal the password of the owner. To make these attacks infeasible and to add an additional layer of security the owner of the funds (who still controls the private key if only his password is stolen) can abort every commit and the person who commited loses his stake. An attacker who steals the password would not commit assuming the owner would abort. Also the owner can change the password by storing a different double hash of a new password.

One important point is the overhead for the smart contract usage. But this overhead is actually quite low. The existing transfer functionality could just remain the same. One just have to add a mapping from addresses to the double-hash of passwords and the functionality to commit and reveal. Also this recovery option can be fully optional. If the user does not store a double-hash of the password onchain all remains as it is today.

My question to you

Is this secure and do you see value in it?