I've been meaning for a few years to type up this simple technique for allowing blockchain users to recover en masse from a future break of ECDSA and other classical algos. Tonight a chance conversation from someone else who is interested in the same problem made me realise that a present-day commitment step in this proposed recovery protocol is unneccesary for anyone currently using an HD wallet (as long as their root secret remains private), and this motivated me to finally type up the full disaster recovery strategy here. The technique is simple and may already be widely known, but I haven't personally read about it, nor had the researcher with whom I was conversing (sorry, I've forgotten your name!), so I thought it worth describing. For all I know I've even done so before myself, but if so I don't know where.

Problem definition:

Some adversary's posession of a quantum computer (capable of breaking the relevant crypto primitives) has become publicly known (techniques to improve the chance of this detection may be described in a later post). We stipulate that some recent point in time (ideally on the order of days at most) can be identified before which most accounts had yet been actively compromised by this adversary. The goal is to roll back the blockchain state (or some subset of it) to this point in time, deprecate the broken primitives, and provide the vast majority of accounts with some safe process for migrating their accounts/assets/etc. to some quantum secure replacement. Note that other accounts and affected parties may not be covered by this plan. Note also that this is not the scenario where the quantum computer has been successfully kept a secret, only the scenario where the existence of the threat has become publicly obvious.

Design goals:

- We want to have a disaster recovery strategy which would apply if this threat were discovered today, not at some point in the future.
- We want the strategy to NOT be inherently committed to a particular quantum-secure replacement algorithm, as current algorithms have many tradeoffs and are being improved and becoming better understood all the time (thus it is hard to predict which algorithm would be most desirable to switch to in advance).
- We want to require as little immediate action from users as possible in order to become eligible for disaster recovery, whether in the present, during the disaster recovery process itself (once it becomes necessary), or even many years after the event.

Technique:

Despite the fact that most Ethereum accounts reuse the same keys many times, and even assuming that a quantum adversary has already obtained the private keys corresponding to every public key ever published on the Ethereum (or any other) blockchain, we can still differentiate between adversary and original user in most cases. To do this we use any (now compromised) public/private key which was derived from an HD wallet as a quantum-secure public commit for a private secret which that user still retains (any node in the HD structure which is "behind" a hardened derivation step will do, up to and including the root secret). We then hard-fork the chain, and institute a simple protocol (such as a commit-and-reveal protocol) allowing the holder of this secret to prove that they have selected a replacement quantum-secure public key (the adversary will not be able to do this step). Once the protocol is completed for a specific user they may continue using their original account with the quantum-secure key. Users may take as long as they wish to upgrade their key, as the adversary cannot initiate the protocol in the meantime. Any compromise which was not actively performed prior to the identified "reset point" will be permanently prevented, even for users who remain offline during this process and indefinitely into the future.

An example, very gas efficient commit-and-reveal protocol works as follows; first the user selects a quantum-secure replacement algorithm, and generates a new public key for it. They then "claim" their insecure account by submitting a quantum-secure on-chain private commitment to BOTH their HD wallet secret (which is different than their account key and not possessed by the adversary) and the new public key (for example, they could submit a hash of the public key and the secret's concatenation). Unless the user has another account which is already quantum secure, this claim will need to be submitted on their behalf by someone else (at this point it has not been authenticated, only recorded). We wait enough time (enforced by the claiming contract) to ensure that the adversary cannot revert the chain beyond the time the commitment was made (we assume that chain consensus has already been made secure against the quantum adversary, potentially by public bootstrapping of the PoS validators in this same fashion). Having ensured that the front-running risk is eliminated, the user submits a quantum-secure proof showing:

a) that their commmitment contained both their secret and a quantum-secure public key, which they then reveal, and

b) that an HD account derivation proceeding from the committed secret node results in the public/private keypair corresponding to the claimed account

This could be as simple as revealing the node and public key and providing an HD derivation path. The contract could verify both that the preimage matches the hash and that the node rederives the original account being claimed, then set the key to be the new owner of the account. Alternatively many other schemes (including ZK-STARKs) can be utilised.

That's it! I haven't really tried to specify and optimise this in detail, I just wanted to mention it. Probably you would want to merkelize the deprecated accounts/keys and then hotwire the ecrecover functionality of the evm to use the replacement algo (plus a lookup of the original pubkey) after an account is properly upgraded, throwing an exception if it hasn't been claimed yet, or something like that. Definitely deploying effective quantum honeypots would increase the chances of noticing

the adversary, thereby minimising the number of parties who get compromised before an acceptable reset point. And of course there are many related techniques that can improve and expand this disaster recovery scheme for users of other chains, etc.

TL;DR:

there is a disaster recovery path that could save most user accounts from being usurped by a quantum-computer-wielding adversary, provided the users are using HD accounts, and we recognise that the adversary exists, key re-use notwithstanding.