So far, I've only seen solutions on ethresearch to quantum proof Ethereum via new keypair types. However, I think there's a more robust solution to migrate Ethereum than hardforking to a quantum resistant keypair, as this would break every single wallet and piece of key-related infra. I think there's a way to quantum-proof Ethereum on the existing ECDSA on secp256k1. The reason Ethereum is not currently quantum proof is that after sending a tx, an account's public key is revealed (i.e. the hash preimage of your address), so an adversary can take the discrete log efficiently with a quantum computer, and get someone's secret key. If there was a way to send txs that didn't reveal the public key, this would allow existing keypairs to remain quantum secure. If you want a brief refresher on what exactly quantum computers can do cryptography wise, you can refer to this blog post/zkresearch post.

So a post-quantum Ethereum account could keep their public key hidden, and only make their addresses public. Then, to send a tx, instead of signing it, they publish a zk proof of knowing a valid signature that corresponds to their address, and that would authorize the transfer, so no one would ever even know their public key! With account abstraction-type solutions, this type of thing could even be possible as soon as that is available on any L2 or L1. It wouldn't work on accounts that have already sent any tx's today (since those reveal public keys), but such accounts could easily send all their assets to a new keypair, and vow to not reveal their public key in those cases. It would quantum proof Ethereum in the long term as well (similarly to how unused utxos in btc are safe right now).

You'd have to use a STARK, since SNARKs right now don't have post-quantum soundness, and the ECDSA proof would have to be fast to generate and verify.

One issue is that smart contracts need to be special-cased, since we know the pre-image of the address via create2. One easy solution is to hard-code that once a contract has been made by create/create2, transactions that utilize their secret key are disallowed (i.e. no signatures or eoa-style txs will be validated).

Perhaps, for future smart contracts, if we don't want to special case them, we could standardize around a new opcode (say create3, or create2 with an optional arg), that, say, just swaps the last bit in the create2 output. This keeps the address determination deterministic, but does not reveal the pre-image of the hash.

This also probably requires some renaming of terminology; for instance, public keys shouldn't be public, so renaming them to be like, quantum insecure keys, hidden keys, or private addresses or something might be useful. It also requires hardware wallets to be able to calculate proofs, which doesn't seem feasible right now – hopefully this usecase drives research for stark computation in low resource environments though. It can also be punted to software for now, as long as the user trusts that the software won't reveal their public key to anyone.